Consultative Committee for Space Data Systems

REPORT CONCERNING SPACE DATA SYSTEMS STANDARDS

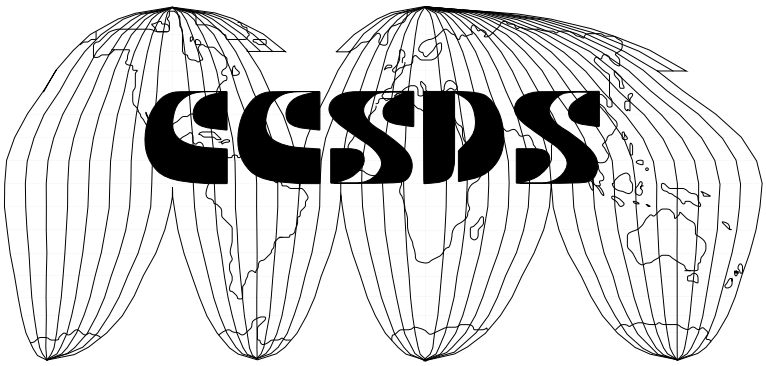
ADVANCED ORBITING SYSTEMS, NETWORKS AND DATA LINKS:

FORMAL DEFINITION OF CPN PROTOCOLS, METHODOLOGY AND APPROACH

CCSDS 705.0-G-2

GREEN BOOK

October 1993



AUTHORITY

Issue:	Green Book, Issue 2
Date:	October 1993
Location:	Vienna, Austria

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and reflects the consensus of technical panel experts from CCSDS Member Agencies. The procedure for review and authorization of CCSDS Reports is detailed in reference [1].

This document is published and maintained by:

CCSDS Secretariat
Program Integration Division (Code OI)
National Aeronautics and Space Administration
Washington, DC 20546, USA

FOREWORD

This document, which is a Technical Report prepared by the Consultative Committee for Space Data Systems (CCSDS), contains explanatory material on the programme of validation applied to the protocols and services defined in the CCSDS Recommendation for *Advanced Orbiting Systems, Networks and Data Links: Architectural Specification*, Reference [2].

Through the process of normal evolution, it is expected that expansion, deletion or modification to this document may occur. This Report is therefore subject to CCSDS document management and change control procedures which are defined in Reference [1].

At time of publication, the active Member and Observer Agencies of the CCSDS were

Member Agencies

- British National Space Centre (BNSC)/United Kingdom.
- Canadian Space Agency (CSA)/Canada.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- Centre National d'Etudes Spatiales (CNES)/France.
- Deutsche Forschungsanstalt für Luft- und Raumfahrt e.V. (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- National Aeronautics and Space Administration (NASA HQ)/USA.
- National Space Development Agency of Japan (NASDA)/Japan.

Observer Agencies

- Australian Space Office (ASO)/Australia.
- Austrian Space Agency (ASA)/Austria.
- Belgian Science Policy Office (SPO)/Belgium.
- Centro Tecnico Aeroespacial (CTA)/Brazil.
- Chinese Academy of Space Technology (CAST)/China.
- Communications Research Laboratory (CRL)/Japan.
- Danish Space Research Institute (DSRI)/Denmark.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Hellenic National Space Committee (HNSC)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Industry Canada/Communications Research Center (CRC)/Canada.
- Institute of Space and Astronautical Science (ISAS)/Japan.
- Institute of Space Research (IKI)/Russian Federation.
- KFKI Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- MIKOMTEK: CSIR (CSIR)/Republic of South Africa.
- Ministry of Communications (MOC)/Israel.
- National Oceanic & Atmospheric Administration (NOAA)/USA.
- Swedish Space Corporation (SSC)/Sweden.
- United States Geological Survey (USGS)/USA.

DOCUMENT CONTROL

Document/Title	Date	Status and Substantive Changes
CCSDS 705.0-G-1 Report Concerning Space Data Systems Standards. Advanced Orbiting Systems, Networks and Data Links: Validation of CCSDS Principal Network Protocols	January 1991	Original Draft Issue. Based on AOS Blue Book issue 1.
CCSDS 705.0-G-2 Report Concerning Space Data Systems Standards. Advanced Orbiting Systems, Networks and Data Links: Validation of CCSDS Principal Network Protocols	October 1993	Current Issue. Editorial changes only; supersedes CCSDS 705.0-G-1.

CONTENTS

Sections **Page**

Figures

REFERENCES

[1] *Procedures Manual for the Consultative Committee for Space Data Systems*. CCSDS A00.0-Y-6. Yellow Book. Issue 6. Washington, D.C.: CCSDS, May 1994 or later issue.

[2] *Advanced Orbiting Systems, Networks and Data Links: Architectural Specification*. Recommendation for Space Data Systems Standards, CCSDS 701.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, November 1992 or later issue.

[3] *Information Processing Systems—Open Systems Interconnection—LOTOS—A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour*. ISO 8807. Issue 1. Geneva: ISO, 1989.

[4] *Advanced Orbiting Systems, Networks and Data Links: Abstract Data Type Library—Addendum to CCSDS 701.0-B-2*. Recommendation for Space Data Systems Standards, CCSDS 705.1-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 1994 or later issue.

[5] *Advanced Orbiting Systems, Networks and Data Links: Formal Specification of the Path Service and Protocol—Addendum to CCSDS 701.0-B-2*. Recommendation for Space Data Systems Standards, CCSDS 705.2-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 1994 or later issue.

[6] *Advanced Orbiting Systems, Networks and Data Links: Formal Specification of the VCLC Service and Protocol—Addendum to CCSDS 701.0-B-2*. Recommendation for Space Data Systems Standards, CCSDS 705.3-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 1994 or later issue.

[7] *Advanced Orbiting Systems, Networks and Data Links: Formal Specification of the VCA Service and Protocol—Addendum to CCSDS 701.0-B-2. Recommendation for Space Data Systems Standards, CCSDS 705.4-B-1*. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 1994 or later issue.

[8] *Architectural Design Document for the CCSDS Validation Programme Software*. Report Logica LOG 212.20213.01-1. Issue 1. Surry, U.K.: Logica Space and Communications Limited, February 1991 or later issue.

[9] *Detailed Design Document for the CCSDS Validation Programme Software*. Report Logica LOG 212.20213.02-1. Issue 1. Surry, U.K.: Logica Space and Communications Limited, February 1991 or later issue.

[10] *Software Users Manual for the CCSDS Validation Programme Software.* Report Logica LOG 212.20213.03-1. Issue 1. Surry, U.K.: Logica Space and Communications Limited, February 1991 or later issue.

1 BACKGROUND AND SCOPE

The production of standards is a process requiring investment of time and money. To ensure that this investment is justified, the standards produced should be unambiguous, consistent, complete and error free, capturing all the original design aims in a fashion which makes them clear to implementors. If the standards are written in natural language, attaining these goals is extremely difficult.

The Validation Programme builds on the many man years of effort involved in the production of the CCSDS Recommendation for Advanced Orbiting Systems (AOS) (reference [2]). Taking the natural language specifications from that document as a starting point, formal specifications were produced and subjected to rigorous analysis and testing, followed by an implementation exercise. The main goals of the Validation Programme were to increase confidence in the AOS Recommendation and to provide implementors with a much higher degree of guidance than would be available from a specification written purely in natural language. The formal specifications are not intended as a replacement for the natural language specifications, but as unambiguous expressions of those specifications, which may be used to clarify any problem areas.

The Validation Programme was initiated at the April 1989 CCSDS Panel 1 meeting as a joint programme involving the MITRE Corporation (working on behalf of NASA) and Logica Aerospace and Defence (working on behalf of ESA). The task of producing formal specifications was divided between the Agencies, and both Agencies undertook to provide implementations of the protocols for interoperability testing. A project carried out by ESA had previously assessed a number of Formal Description Techniques, selecting the International Standardisation Organisation (ISO) language LOTOS (reference [3]) and applying it to a subset of the AOS Recommendation. As a result of this project, LOTOS was selected for use on the Validation Programme.

The final outputs of the Validation Programme are:

- Formal specifications of the AOS Protocols and Services.
- Formally specified Test Suites for the AOS Protocols and Services.
- Reference implementations of the AOS Protocols.
- A set of issues concerning the CCSDS AOS Recommendation.

The purpose of this report, therefore, is to describe the development of the Validation Programme over its eighteen-month lifetime, to capture and relate the experience gained in the use of LOTOS, to document the results of the programme in terms of both products and problems found within the original (natural language) specification, and to make available the lessons learned during the specification and implementation of the CCSDS AOS protocols.

2 METHODOLOGY

The specification methodology used by the CCSDS in reference [2] adheres to the layered architecture developed by ISO for the Open Systems Interconnection standards; communications protocols and the services they provide are described in a formalised manner. Both ISO and CCSDS have recognised that although this level of formalism is of value in the production of standards, the use of English is still a basic problem, leading to ambiguities, omissions, inconsistencies and errors which are undetectable by any automated method. These problems are of particular note for CCSDS, whose Recommendations are used by a number of Space Agencies in the formulation of standards; misinterpretations will result in non-interoperation of projects, undermining cooperation between Agencies and diminishing confidence.

The layered approach used in reference [2] led to the identification of three major protocol layers:

- *Path Layer*. This communications layer transfers variable-length data units end to end across a network. Note that this network may consist of many subnetworks; for example, a network may consist of a Local Area Network (LAN) on board a spacecraft, a Radio Frequency link to a ground station, and one or more ground-based LANs.

- *Virtual Channel Link Control (VCLC) Layer*. This communications layer transfers variable-length and continuous data over space-ground and space-space communications channels. It provides several different services, all based around the concept of 'Virtual Channels': logical-channel subdivisions of a physical space channel.

- *Virtual Channel Access (VCA) Layer*. This layer transfers fixed-length data units over space-ground and space-space communications links. It is used by the VCLC, and deals directly with the physical space channel to support the Virtual Channel abstraction. The VCA contains (and thus hides) functions such as bit synchronisation, error detection and correction, and retransmission, which are needed to operate the higher-level layers over Radio Frequency links.

The Space Link ARQ (Automatic Repeat Queuing) Procedures (SLAP) were neither specified nor implemented, as the English Language specifications were not deemed sufficiently mature. The Insert procedures were specified in LOTOS but were not implemented, as their orientation towards isochronous data transfer made them unsuitable for the Validation Programme implementations. The (255,223) Reed-Solomon forward error correction routines were not implemented, as their performance is well-known and does not require validation.

From this starting point of formalised English Language specifications, the Validation Programme proceeded to the production of two LOTOS specifications for each layer. Of these, one specification dealt with the services provided by that layer to its superior layer(s), and the other concentrated on the protocols involved in the provision of those services.

The LOTOS specifications were then subjected to rigorous testing at a number of levels. In the first instance, syntactic and semantic analysers were used to check that the specifications were compliant with the LOTOS standard (reference [3]). In the next phase of testing, a simulation tool was used to animate the specifications, and their behaviour was examined and evaluated. Finally, a set of test processes was written in LOTOS, expressing the behaviour required of the protocols and services under certain operating conditions. Using a specially developed tool, these test processes could be combined with the LOTOS specifications to form a new, combined specification. These new specifications were then subjected to further simulation.

The production of these LOTOS specifications proved to be a very valuable phase of the Validation Programme in terms of detecting errors. The process of formalising the English Language specifications itself led to the identification of many ambiguities and insufficiencies in the original issue of the main Recommendation. The main areas of improvement in terms of producing the formal specification were in clarifying the interface descriptions (e.g., convergence layers were identified for Path and the 8473 protocol to work over VCLC), ensuring consistent use of addressing information, and identifying the management information required for the operation of the protocols. The preparation of tests to be applied to the LOTOS specifications highlighted some further omissions in the original specifications. (The problems identified have been corrected in the current issue of the main Recommendation.)

When the formal specifications had been developed to the satisfaction of both ESA and NASA, a process of implementation was initiated. Each party produced implementations of the CCSDS AOS protocols derived from the LOTOS specifications and complying with the agreements discussed in section 4.

In addition to the protocol implementations, test suites were derived from the LOTOS test processes described above. These test suites were used by both ESA and NASA to test their implementations locally; a further period of testing involving the interoperation of ESA and NASA implementations was planned, but was not completely accomplished within the Programme.

The Validation Programme methodology is summarised diagrammatically in Figure 2-1. At each stage feedback was provided to the previous stages; in practice, however, the formal specification exercise was the only stage to uncover any notable errors. The implementation and testing exercises proceeded without the discovery of any errors which required modifications to the specifications.

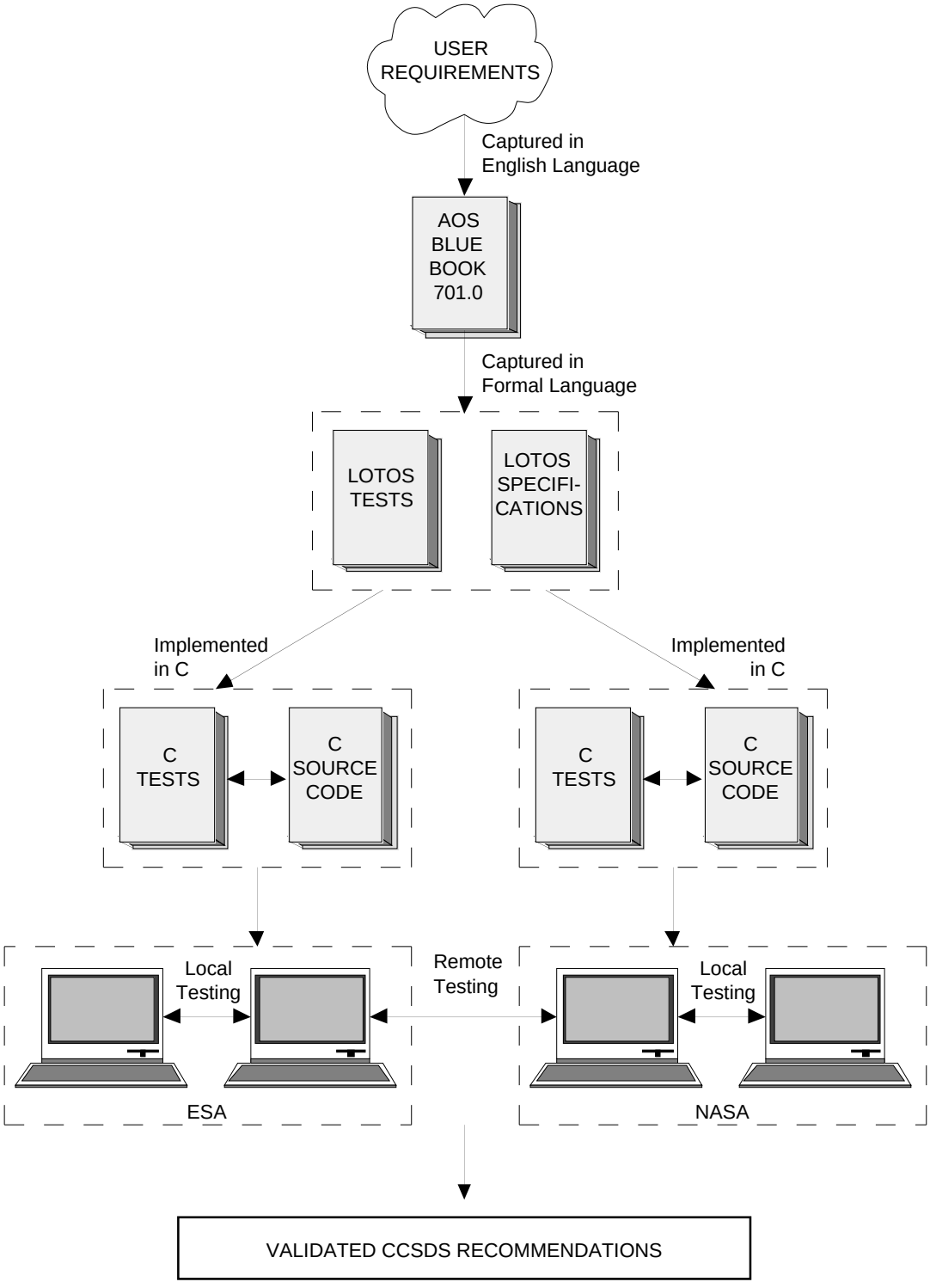


Figure 2-1: Validation Process

3 LOTOS APPLICATION AGREEMENTS

Formal Description Techniques are not yet widely used within Information Systems development. With the establishment of the LOTOS standard within ISO, however, a rising recognition of the value of these techniques is signalled. The Validation Programme is one of the first projects to use LOTOS, and thus much valuable experience has been gained in the application of such a formal method to the specification and development of communications systems.

LOTOS is an extremely flexible and expressive language, with all the great advantages that these qualities bring. With these qualities, however, come certain disadvantages: there may be many different methods of expressing a certain concept, styles may arise which differ considerably, and so on. In order to obtain maximum benefit from inter-agency cooperation, it is not sufficient simply to use the same formal specification language; agreements must exist concerning how the language is used, how certain concepts are expressed, and so on. This section outlines the agreements reached by ESA and NASA concerning these subjects.

For an introduction to the LOTOS language, readers are referred to the tutorial section of reference [3]; a basic knowledge of LOTOS will be required to understand this section of the Report.

3.1 LOTOS STYLES

Distinguishing styles in LOTOS is a far from exact science. The following four styles are, however, widely accepted as being prevalent. It should be noted that LOTOS specifications often exhibit properties of more than one style. The major differences exist between the ‘constraint-oriented’ style and the other three which are often applied in concert.

The Validation Programme, having relatively detailed English Language specification as input, produced specifications which are closest to the Monolithic and Resource-Oriented styles.

3.1.1 Constraint-Oriented Style

The Constraint-Oriented style consists of specifying logical assertions or properties which the system must satisfy. These assertions or constraints may be grouped together to separate concerns and then combined (as independent threads) to give all the constraints on the system specified.

The Constraint-Oriented style is the most abstract of the four covered here. Its strengths lie in the ability to capture a great deal of information about complex systems in a compact form; to produce a specification which will satisfy system requirements without involving extraneous detail, and to separate concerns, leading to a ‘component engineering’ or object-oriented approach. The criticisms levelled at this style (justifiably) are that it is difficult to extract overall system behaviour (as a result of the separation of concerns), and that it is very abstract, thus leading to specifications which are difficult to implement. The last two criticisms are a result of LOTOS’s being used as a high-level specification language; the Constraint-Oriented

specification must be refined and expanded into one of the three following forms to be useful to implementors.

3.1.2 Resource-Oriented Style

The Resource-Oriented style, as its name suggests, concentrates on specifying the resources which the system in question must manage. For example, a Resource-Oriented specification of the OSI Transport Layer would focus on the connections which the layer must administer.

The Resource-Oriented style is much less abstract then the Constraint-Oriented style. In the latter we would specify properties such as the relative ordering of connection requests and data transfers; in the former we would concentrate on resources such as network connections, their properties, and the ways in which they may be used to support the Transport Layer service.

Implementation of a Resource-Oriented specification is far simpler than implementation of a Constraint-Oriented specification, but still may require that a certain amount of refinement be carried out.

3.1.3 State-Oriented Style

The State-Oriented style is perhaps the simplest and least rewarding of those covered here. Often State-Oriented specifications consist of just one process whose behaviour is controlled by one or more state variables.

3.1.4 Monolithic Style

The Monolithic style leads to a relatively explicit specification of data structures, processing algorithms and system behaviour. Implementations may be derived from Monolithic specifications with relative ease, this process consisting mainly of ‘concretising’ the Abstract Data Types and system interfaces and implementing the system behaviour; i.e., little or no refinement or expansion should be necessary. The Monolithic style is effectively an abstract implementation.

3.2 SPECIFICATION LIST

The CCSDS AOS Blue Book (reference [2]) specification was partitioned into the following LOTOS Specifications; the numbers to the right are the CCSDS document references for the Recommendations containing the LOTOS Specifications.

ADT Library	705.1
Path Service	705.2
Path Protocol	705.2
Path-to-Multiplexing Convergence Layer	705.3
8473-to-Encapsulation Convergence Layer	705.3
VCLC Service	705.3
VCLC Protocol	705.3
VCA Service	705.4
VCA Protocol	705.4

3.3 LOTOS INTERFACES

The LOTOS specifications use ‘gates’ for events which involve transfer of information. Gates may be either exposed or hidden, the former representing external interfaces, the latter representing internal interfaces. Exposed gates are used for each service (provided or used) and for interaction with management. Hidden gates are used for various purposes, such as communications between the Encapsulation procedures and the Multiplexing procedures within the VCLC (as in Figure 3-1, below).

The gates exposed by each Specification are as follows:

– Path exposes four gates: two for the services it offers (Packet and Octet String), one for the service it makes use of (Subnetwork), and one for management.

– The Path-to-Multiplexing Convergence layer exposes two gates: one for Path to use and one for the Multiplexing service.

– The 8473-to-Encapsulation Convergence layer exposes two gates: one for 8473 to use and one for the Encapsulation service.

– The VCLC exposes seven gates: three for the services it offers (Multiplexing, Encapsulation, and Bitstream), one for the service it makes use of (VCA), two for release events (directing the protocol when to release data units for the Multiplexing and Bitstream services), and one for management.

– The VCA exposes seven gates: three for the services it offers (VCA, Insert, and VCDU), one for the service it makes use of (Physical Channel), one for release events (directing the protocol to release data units), one to determine the bit rate at which data should be transmitted, and one for management.

3.4 SERVICE INTERFACE STYLE

The primitives used to exchange information at the service interfaces (gates) involve two types of information. The Service Data Units (SDUs) and Protocol Data Units (PDUs) are distinguished from addressing information, such as a Virtual Channel Identifier. Each primitive involves both the Data Units and the addressing information.

As the LOTOS specifications contain multiple protocol or service entities, each dealing with one set of address information (such as one Virtual Channel), a mechanism is required to indicate which address a particular entity is using.

The mechanism we have chosen is to have both senders and receivers synchronize on the address information and exchange the Data Units. In Figure 3-1 below there are three Multiplexor entities active dealing with Virtual Channels 23, 25 and 52; any Multiplexing request primitives on Virtual Channel 25 arriving on the 'MUX' gate will thus be dealt with by the middle entity in the Multiplexing Procedures block.

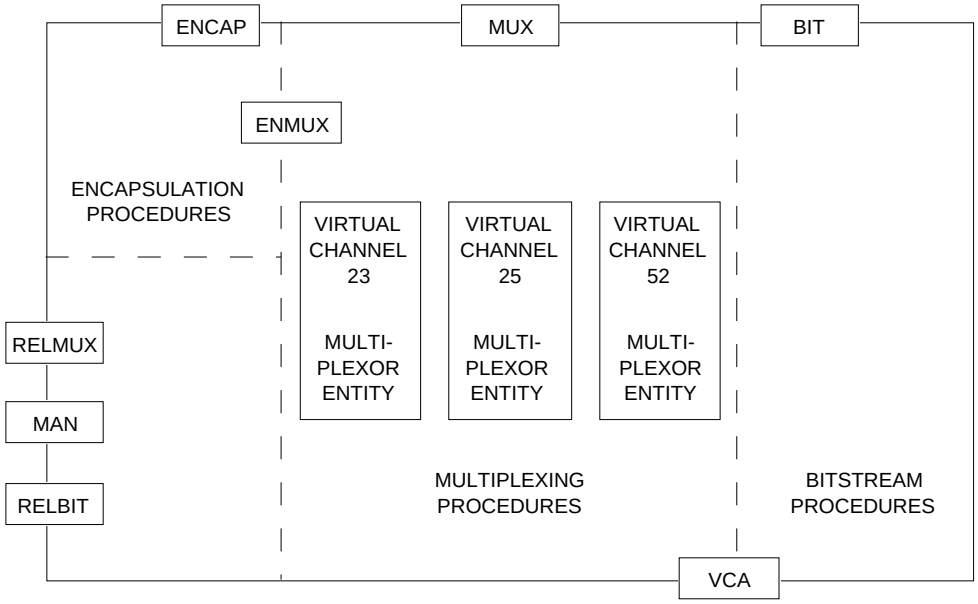


Figure 3-1: VCLC Gates

Where some or all of the addressing information is contained in a Data Unit, the specifications highlight the addressing information as a separate parameter of the primitive. The main Recommendation does not currently do this. For instance, the VCA_VCDU.request in the main Recommendation consists solely of the VCDU/CVCDU, since the addressing information is the VCDU-ID, contained in the Data Unit. However, in the LOTOS specifications, the VCA_VCDU.request appears as:

```
Sending_Gate      ! VCDU/CVCDU      (* Data Unit *)
                    ! VCDU-ID       (* Addressing Information *)
Receiving_Gate    ? VCDU/CVCDU      (* Data Unit *)
                    ! VCDU-ID       (* Addressing Information *)
```

The important distinction (in LOTOS) between the use of '!' and '?' is:

- If two entities (e.g., layers N and N-1) both have '!' in front of a value, then they will synchronize if and only if the values are identical.

- If one entity has '!' and the other '?', then the 'variable' following the '?' will take on the value following the '!'; thus data will be transferred from the sending process to the receiving process.

The distinction between the case where both sender and receiver have '!' and the case where the sender has '!' and the receiver '?' is, therefore, the distinction between synchronization and data passing.

This convention does not mean that the addressing information contained in the Data Unit would be transmitted twice. It would only be transmitted once. This convention merely serves to highlight the addressing information; it does not duplicate information.

The VCLC to VCA Service Interface is slightly more complex than the other service interfaces, as the VCA_SDU may be either an M_PDU, a B_PDU, or a SLAP_PDU. However, the VCA does not need to know the internal structure of the VCA_SDU. The LOTOS specification of the VCLC protocols passes all PDUs to the VCA as OctetStrings, and expects all PDUs received from the VCA to be OctetStrings.

3.5 MANAGEMENT INFORMATION INTERFACE

For the protocols to be able to operate, they need a minimum set of management information. The LOTOS specifications produced have three categories of management information.

1) *Implementation-Specific Information*. This information will be the same for all the protocol entities in a specific implementation (for example, the fill pattern to be used in idle data units). This information will be represented as the formal parameters to the specification.

   ```
   specification VCLCProtocol [gates] (formal parameters): noexit
   ```

2) *Protocol Entity Configuration Information*. This information is used to configure specific protocol entities. It will be represented as information received from the management gate.

```
man    ? PathID
       ? ServiceType
       ... ;

NewProtocolEntity [gates] (PathID, ServiceType, ...)
```

3) *Table Lookup Information.* This information is not necessary for configuring the protocol entity, and may change during the operation of a protocol entity. It will be represented as information requested by protocol entities and received over the management gate.

```
man    ! VCDU_ID
       ? VC_PDU_Length
       ? Physical Channel
       ...
```

Section 5.5.1.2 of the main Recommendation (reference [2]) specifies some additional management information which defines the conditions under which the VC_PDU is to be released for transmission.

The LOTOS representation for these release parameters is for the VCLC and VCA layers to present distinct timer gates; events on these gates direct the layers to release data units.

3.6 MANAGEMENT PRIMITIVES

Any specification of management primitives is outside the scope of the LOTOS specifications listed in section 3.1. Only management information as discussed in section 3.4 will be in the LOTOS specifications.

3.7 SCOPE OF LOTOS SPECIFICATIONS

Each LOTOS specification will deal with either the protocol or the service provided by the layer. Each specification may thus instantiate multiple protocol or service entities; for example, Figure 3-1 above shows three Multiplexing Protocol Entities which will have been instantiated in response to the directives received from the ‘MAN’ gate.

3.8 IDENTIFICATION OF MANAGED INFORMATION

All the managed information specified according to section 3.4 has been collated and categorised in section 4.5. This document will serve as a basis for analysing the management information present in the LOTOS specifications. After analysis of this information, the information will be fed into the Management and Signalling Recommendations.

This set of managed information indicates the minimum needed for the formal specification of the protocols. There will undoubtedly be some extensions to this set which are directly relevant to implementations of the protocols.

3.9 PROTOCOL SPECIFICATION GUARDS

The LOTOS specification of a protocol requires data exchange over gates to have a certain format, e.g., the ordering of the data items passed and their types. It is possible in LOTOS to place 'guards' at the gates which further restrict events happening at those gates.

The use of these guards has been limited to examining the Data Units being passed. These guards only check fields in those Data Units (such as Version Numbers) to ensure that they can be correctly processed by the Protocol or Service specified.

Guards are not used for checking addressing information or for checking the consistency of management information.

3.10 LOTOS AND MAIN RECOMMENDATION ALIGNMENT

As far as possible, LOTOS specifications should be aligned with the main Recommendation. Specifically, alignment includes:

- LOTOS process descriptions should match the main Recommendation procedures, both in name and functionality;

- LOTOS interactions at gates should match the main Recommendation primitive specifications, both in parameters and their order.

3.11 SERVICE SPECIFICATIONS

The Service Specifications deal with the end-to-end performance provided by the communications protocols; thus a single Service Specification may represent multiple 'protocol machines' as shown in Figure 3-2:

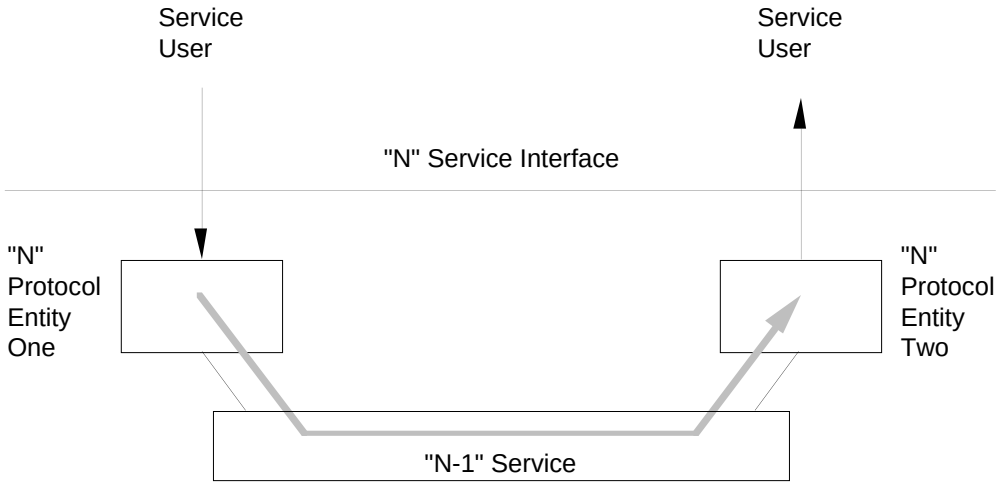


Figure 3-2: Service Specification

Note that the Specification will provide only one gate per service; thus, externally a user request would be represented as

```
gate    ! VirtualChannel
        ! Data
```

and the corresponding service indication at the destination would be

```
gate    ! VirtualChannel
        ! Data
```

(i.e., they would be identical).

To avoid confusion, the LOTOS Service specifications instantiate service entities, each having a Source process which only receives data using the exposed gate(s), and a Destination process which only sends data using the exposed gate(s). They are linked using a hidden gate. In the case of the Path Service, with multiple Destination processes possible, the multiple Destination processes would all communicate via a hidden gate. Even though the addressing information is the same when received as when delivered, this approach makes explicit that the source and destination are different.

3.12 BITSTREAM INTERFACE

The Bitstream interface is represented in the LOTOS specification as transferring one bit at a time.

The AOS Blue Book (reference [2]) states that, “Since the service interface specification is an abstract specification, the implementation of the Bitstream Data parameter is not constrained; i.e., it may be continuous Bitstream, delimited Bitstream, or individual bits” (reference [2], 5.3.7.5.e).

Representing a continuous (i.e., potentially infinite) bitstream is impossible in LOTOS; the approach taken in the LOTOS specification was to transfer individual bits over the bitstream interface and build them into Bitstream Protocol Data Units in the Bitstream Procedures. Similarly, at the termination of the Bitstream Virtual Channel, the Bitstream Protocol Data Unit is dissembled into a string of bits which are passed over the Bitstream interface one at a time.

3.13 CONVERGENCE

The Subnetwork service, which Path requires as a lower-layer service, does not match the Multiplexing service provided by the VCLC. Therefore, a Path-to-Multiplexing Convergence Layer has been specified in LOTOS for the purpose of transforming the output of the Path (which in the case of using the Space Link Subnetwork will have a <PCID, VCDU-ID> pair as both Source and Destination address, since it indeed acts as both) into the format expected at the Multiplexing service of the VCLC.

The output of the 8473 protocol does not exactly match what is expected at the Encapsulation service provided by the VCLC. Therefore, an 8473-to-Encapsulation Convergence Layer will be specified in LOTOS for the purpose of transforming the output of the 8473 layer into the format expected at the Encapsulation service of the VCLC.

4 IMPLEMENTORS' AGREEMENTS

4.1 HARDWARE AND SOFTWARE

The hardware platform selected for the Validation Programme implementation was the Sun UNIX Workstation. The programming language used was C, with TCP/IP used as the Inter-Process Communication (IPC) mechanism.

The hardware configuration used for local testing of the CCSDS implementations is shown in Figure 4-1:

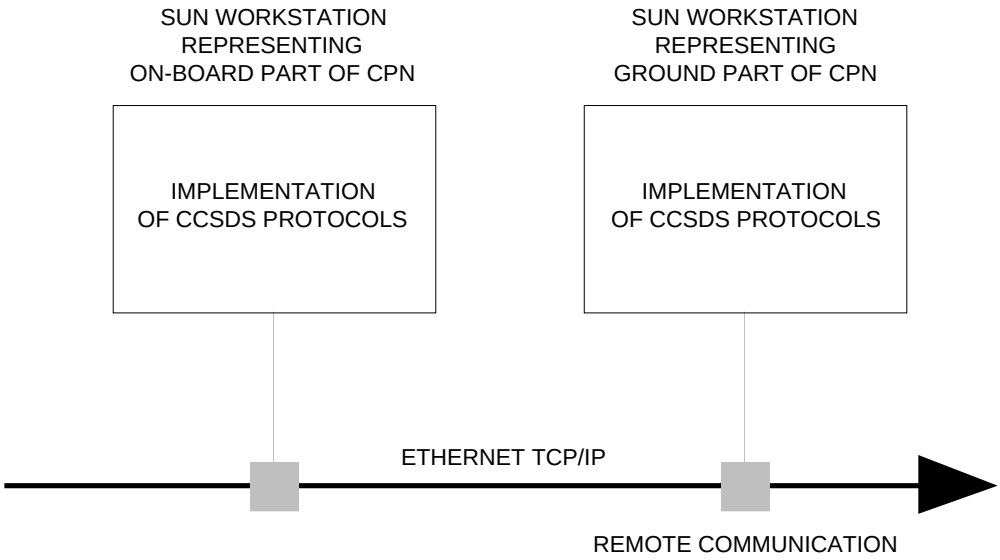


Figure 4-1: Local Testing Hardware Configuration

For interoperability and remote testing, the Ethernet LAN is effectively replaced by a combination of the Trans-Atlantic Link and the associated networks which allow communication to take place between the Sun Workstations in ESTEC and those in NASA.

4.2 ARCHITECTURE

Central to the Validation Programme implementations is the concept of a Virtual Node: the protocol layers are grouped into a Virtual Node which provides data transfer services to other processes. These Virtual Nodes may be interconnected by a generic Subnetwork Service or a Physical Channel Service (see below). This grouping allows great flexibility when running the implementations: several Nodes may execute on a single Sun Workstation, one Node may run per Workstation, or the constituent layers within a Virtual Node may be distributed over several Workstations.

Figure 4-2 below shows a possible Virtual Node configured as a gateway between an on-board Local Area Network and the Space Link Subnetwork. The shaded blocks represent interfaces between layers (and exposed interfaces). The interface between the Path Layer and the VCLC Layer is the Path-Multiplexing Convergence Layer Interface. The interface between the VCLC Layer and the VCA Layer is the VCA_UNITDATA service. Note that the Node Manager has private interfaces (i.e., not available for other user entities) to the Path, VCLC and VCA Layers, but does not manipulate the Subnetwork Layer.

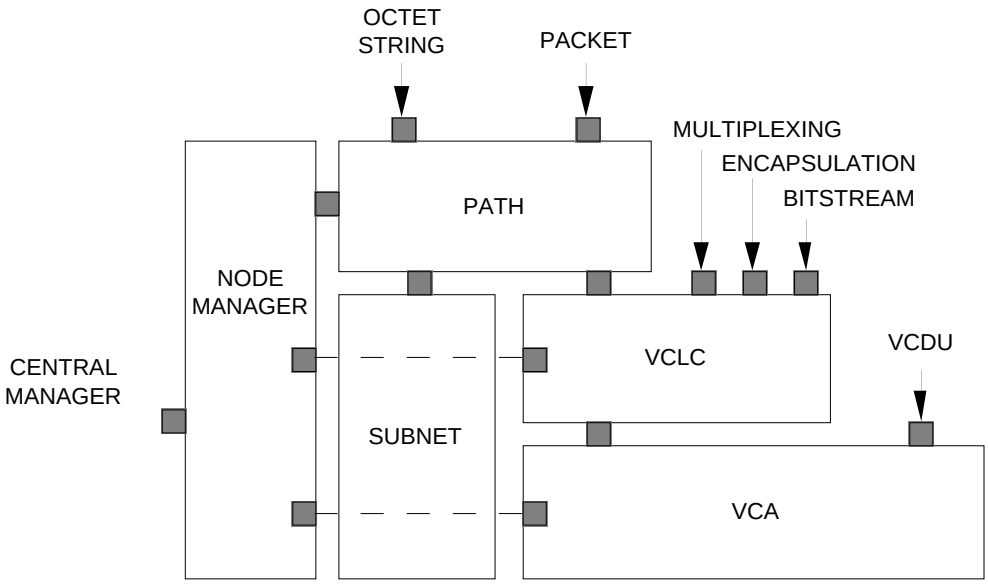


Figure 4-2: Virtual Node Configured As Gateway

The functions of the Path, VCLC and VCA Layers are fully covered in references [2], [5], [6] and [7]; the other entities required for the Validation Programme implementations to function and interwork are described below:

4.2.1 Generic Subnetwork

The service expected by the Path from the Subnetwork is defined in reference [2] as being the connectionless-mode transfer of data from a source subnetwork point-of-attachment address to a destination subnetwork point-of-attachment address. The Subnet process implements this service in a generic fashion, providing data transfer to the Path Protocol implementations.

4.2.2 Node Manager

Because of the immaturity of the CCSDS Recommendations for Management and Signalling, neither a Node Manager nor the services required from a Node Manager are specifically identified in reference [2]. In order to operate the protocol implementations, the Validation Programme produced a simple Node Manager which accepts Management Directives from a Central Manager and configures the protocol layers in its Virtual Node accordingly. Section 4.5 gives more detail on Management interfaces.

4.2.3 Central Manager

The Central Manager maintains a database containing Management Information for all Virtual Nodes under its supervision. It may signal this information to the Node Managers directly supervising each Virtual Node.

4.2.4 Physical Layer

The Physical layer handles up to ten Physical Channels (arbitrary limit), allowing users to register as either receivers or transmitters on these channels; only one transmitter is permitted per Physical Channel, but multiple receivers are allowed. The Physical layer implementation accepts Channel Access Data Units from transmit processes and provides data as unformatted octets to receive processes. The Physical layer permits the injection of noise into Physical Channels (under user control) and supplies randomised noise to any process registered as receivers on Physical Channels with no transmitters.

4.3 SERVICE INTERFACES

TCP/IP was chosen for the Validation Programme Implementations for a number of reasons:

– The service interface has to be reliable; i.e., the probability of a service request's being lost should be minimal (note that this is not the same as a reliable or unreliable service per se; rather it concerns the nature of service primitives which should always be reliably delivered).

– The geographical separation of ESA and NASA make necessary the use of Internet protocols. TCP/IP is the reliable stream transport service layered on top of IP, thus permitting the Agencies to run an 'N'-layer protocol implementation and use the same IPC system to communicate with an 'N-1'-layer protocol implementation, whether it is on the same machine, on a different machine on the same network, or on a different machine on a different (but connected) network.

– TCP/IP implementations are available as standard on the Sun workstations used by both ESA and NASA for the Validation Programme.

The TCP/IP connections used for protocol layer interfaces all share the same basic life cycle illustrated below:

1 The Service Provider Process (SPP) must be running before any Service User Processes (SUPs) are initiated. The SPP creates a number of passive sockets (TCP/IP ports, shown as darkly shaded blocks in Figure 4-3). In each case, the SPP will create a passive socket for each service it provides.

2 An SUP may now make a connection to the SPP's passive sockets using a TCP/IP active socket (shown as a lightly shaded block in Figure 4-3); at this stage the connection may be rejected by the SPP (if, for example, it cannot deal with any further connections).

3 If the SPP accepts the connection, it will then create a new socket and link the SUP with that socket.

4 The SPP now returns to monitoring the passive socket for further connections, but will also deal with any input from the newly created connection to the SUP. Note that this connection is still, in effect, anonymous; a CCSDS Service Access Point (SAP) address will have to be associated with the connection by the SUP as described later.

5 A new SUP may now attempt to connect with the SPP.

6 Steps 2–4 will be repeated for the new connection.

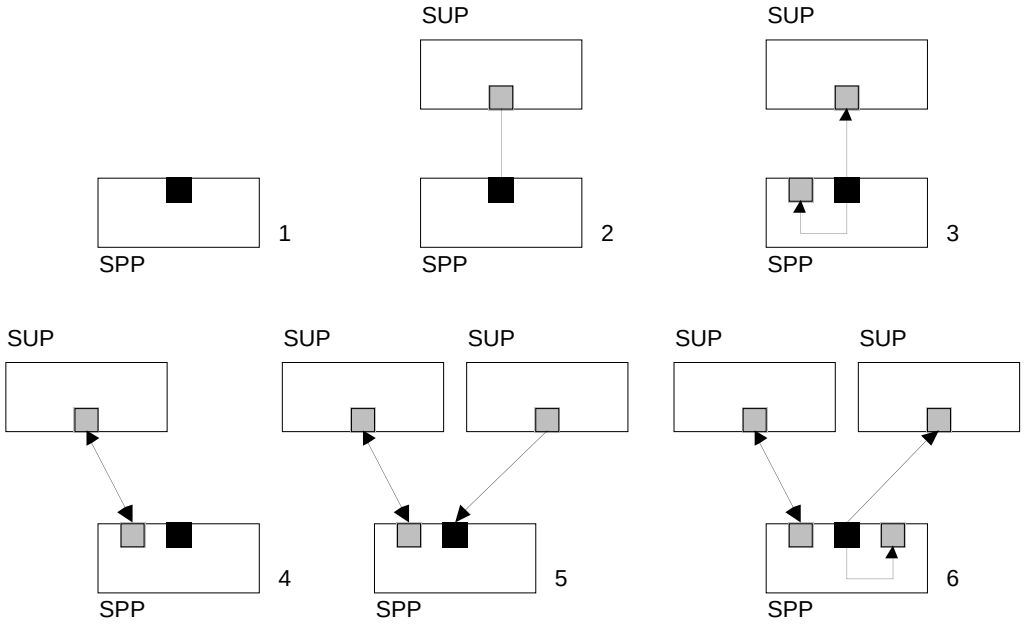


Figure 4-3: TCP/IP Connections for Protocol Layer Interfaces

Note that many CCSDS SAPs may be associated with a single TCP/IP connection. This is achieved by the SUP's passing of a number of SAP Registration Messages (described below) to the SPP.

There is a one-to-one mapping between service user processes and service provider processes within the Virtual Nodes (note, however, that several User Entities may use the services of a single protocol layer within a Virtual Node). Thus, for example, there will be one and only one VCA service provider process for a VCLC process in a particular Virtual Node. This arrangement avoids the need for service provider/user selectors which would otherwise have to be managed parameters.

4.4 MESSAGE FORMATS

This section details the structure of all messages exchanged between the Validation Programme implementations. For ease of implementation, all messages have a six-byte header prefixed containing message length and type information.

4.4.1 SAP Registration Messages

The first set of messages deals with the identification of a Service User Process to a Service Provider Process. Identification is achieved by the user process's registering itself with the provider process using a Service Access Point (SAP) Registration Message.

4.4.2 Primitive Messages

The second set of messages actually provides the format of the protocol primitives.

4.4.3 Management Messages

The final set of messages provides the mechanism for central management facilities to communicate with a management agent and vice versa.

This section covers only the fixed parts of the messages; the variable parts of the management messages are covered in 4.5.

4.4.3.1 MAN_SET

Message Length in bytes	4 Bytes
Message Type	2 Bytes
Managed Object Type	2 Bytes
Variable	N Bytes

4.4.3.2 MAN_START

Message Length in bytes	4 Bytes
Message Type	2 Bytes
Managed Object Type	2 Bytes
Variable	N Bytes

4.4.3.3 MAN_CEASE

Message Length in bytes	4 Bytes
Message Type	2 Bytes
Managed Object Type	2 Bytes
Variable	N Bytes

4.4.3.4 MAN_GET

Message Length in bytes	4 Bytes
Message Type	2 Bytes
Request ID	2 Bytes *
Managed Object Type	2 Bytes
Variable	N Bytes

4.4.3.5 MAN_GET_RESPONSE

Message Length in bytes	4 Bytes
Message Type	2 Bytes
Report ID	2 Bytes*
Managed Object Type	2 Bytes
Variable	N Bytes

* The request ID used in the MAN_GET message will be placed in the report ID field of the MAN_GET_RESPONSE message as an aid to tracking GET/RESPONSE pairs.

Service primitives will be accepted only when the associated Managed Object is in the Active state.

4.5 MANAGEMENT INTERFACE

This section covers the Management entities implemented by NASA and ESA. Each entity will be implemented as a UNIX process using TCP/IP as the interprocess communication system, thus permitting the geographical separation of entity and entity user over different machines and networks.

Two entities have been identified: a Central Manager, having control over many Virtual Nodes (via the Management Agents), and a Management Agent, responsible for the operation of a single Virtual Node (where a Virtual Node is more than one of, and usually all of, Path, VCLC, VCA and/or Subnet). Layer Management Entities, responsible for the management needs of single protocol layers, are private matters for each party. It is worth noting here that this privacy allows that only Party A may directly manipulate the protocol layer implementations belonging to it; Party B can manipulate Party A's layers only indirectly by communicating with Party A's Management Agents.

Thus, for each Virtual Node running, there will also be a Management Agent. The Management Agents communicate information to the protocol layer implementations under their control by private means (i.e., the Management Agent to Layer Management Entity interfaces are not cross supported between ESA and NASA).

The Management Agents are capable of supporting simultaneous connections from two Central Managers. This capability enables (for example) NASA's Central Manager to be constantly connected to its Management Agents without denying access to ESA's Central Manager.

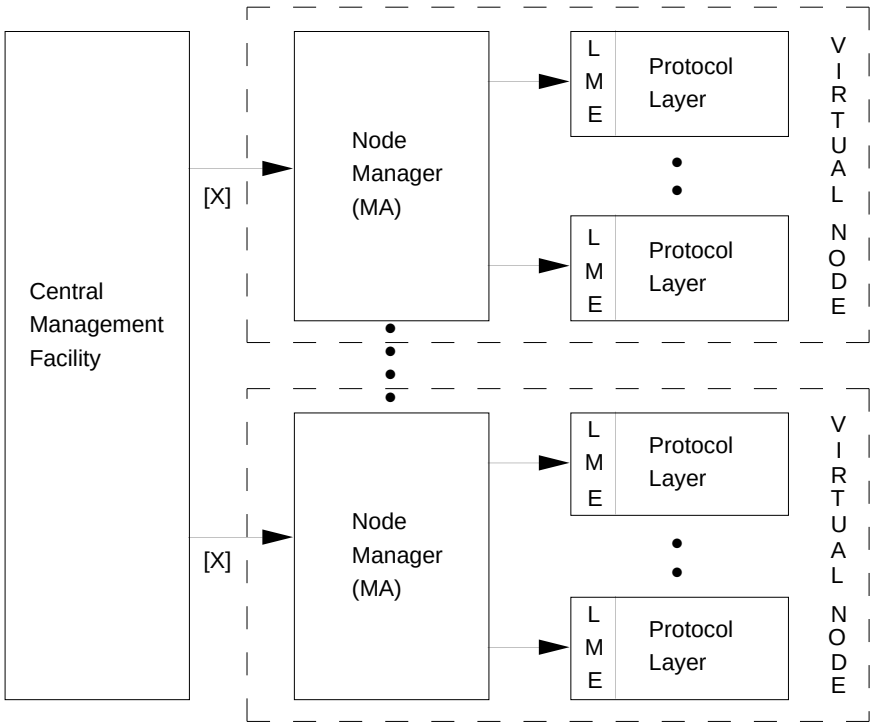
The Management Agent interface is implemented in a manner similar to the service interfaces. A TCP/IP port is exposed by each Management Agent.

The Management Agents are unintelligent; i.e., they configure the protocol layers in direct accordance with the instructions they receive. If a Management Agent is instructed to set up a certain Logical Data Path, running over a certain Virtual Channel, it will correctly configure the Path Layer, but will not set up the required Virtual Channel in the VCLC and VCA layers unless explicitly instructed to do so.

It should be noted that the Layer Management Entity functions are embedded within the Protocol Layers, and as such may be viewed more as the interface between the Management Agents and the Protocol Layers than as separate entities.

Five Management Primitives have been implemented:

Cease Operation	on a Managed Object
Start Operation	on a Managed Object
Set	a Managed Object to certain value(s)
Get	on a Managed Object
Response	a Managed Object's parameters



Key:

[X] – Exposed Interface
MA – Management Agent
LME – Layer Management Entity

Figure 4-4: Management Interface

These message formats have been (as far as possible) aligned with the CMIS/CMIP work carried out in ISO/OSI. All the operations are unconfirmed; thus, to ensure that a Set operation has been successful, the user will have to execute a Get operation and check the resulting Response message. Note that this does not prevent either party from implementing a more intelligent Management Process which could (for example) carry out a Set/Get combination to provide a confirmed, higher level, Set operation.

The MAN_GET, MAN_CEASE and MAN_START messages operate on the Managed Object Type and Identifier; the MAN_SET message operates on the Managed Object Type, Identifier, and Parameters; the MAN_GET_RESPONSE message is sent from the Management Agent to the Central Manager and contains the Managed Object Type, Identifier, and Parameters in addition to the request ID from the original MAN_GET message (see below).

MAN_START and MAN_CEASE

Message Length in bytes	4 Bytes
Message Type	2 Bytes
Managed Object Type	2 Bytes *
Managed Object Identifier	N Bytes

* the Managed Object Identifier will vary from Managed Object to Managed Object.

Thus, to start a Logical Data Path Originator, the message passed from the Central Manager to the Management Agent might be:

12	Length
31	Type (= MAN_START)
1	Object (=Path Originator)
127, 58	APID, Qualifier

MAN_SET

Message Length in bytes	4 Bytes
Message Type	2 Bytes
Managed Object Type	2 Bytes
Managed Object Identifier	N Bytes
Parameters	P Bytes

MAN_GET

Message Length in bytes	4 Bytes
Message Type	2 Bytes
Request ID	2 Bytes
Managed Object Type	2 Bytes
Managed Object Identifier	N Bytes

MAN_GET_RESPONSE

Message Length in bytes	4 Bytes
Message Type	2 Bytes
Response ID	2 Bytes
Managed Object Type	2 Bytes
Managed Object Identifier	N Bytes
Parameters	P Bytes

The Managed Objects upon which these primitives act are listed below:

4.5.1 Path Layer Managed Objects

Name: Path_Originator
Type: 1

Identifier:

APID	2 Bytes
APID Qualifier length in bits	4 Bytes
APID Qualifier	length/8 Bytes

Parameters:

Service	2 Bytes
Number of Remote Terminations	4 Bytes
Remote Terminations	N Bytes

Where a Remote Termination is as follows:

Subnet ID length in bits	4 Bytes
Subnet SAP length in bits	4 Bytes
Subnet ID	length/8 Bytes
Source Subnet SAP	length/8 Bytes
Destination Subnet SAP	length/8 Bytes

Name: Path_Relayer
Type: 2

Identifier:

APID	2 Bytes
APID Qualifier length in bits	4 Bytes
APID Qualifier	length/8 Bytes

Parameters:

Service	2 Bytes
Subnet ID length in bits	4 Bytes
Subnet SAP length in bits	4 Bytes
Subnet ID	length/8 Bytes
Subnet SAP	length/8 Bytes
Termination Flag	2 Bytes
Number of Remote Terminations	4 Bytes
Remote Terminations	N Bytes

Where a Remote Termination is as follows:

Subnet ID length in bits	4 Bytes
Subnet SAP length in bits	4 Bytes
Subnet ID	length/8 Bytes
Source Subnet SAP	length/8 Bytes
Destination Subnet SAP	length/8 Bytes

4.5.2 VCLC Layer Managed Objects

Name: Encapsulation
Type: 3

Identifier:

APID	2 Bytes
VCDU_ID	2 Bytes

Parameters: None.

Name De-Encapsulation
Type: 4

Identifier:

APID	2 Bytes
VCDU_ID	2 Bytes

Parameters: None.

Name: Multiplexing
Type: 5

Identifier:

VCDU_ID	2 Bytes

Parameters:

M_PDU Length in Bytes	4 Bytes
Number of Packet Channel IDs	2 Bytes
Packet Channel IDs	N*2 Bytes

Name: De-Multiplexing
Type: 6

Identifier:

VCDU_ID	2 Bytes

Parameters:

Number of Packet Channel IDs	2 Bytes
Packet Channel IDs	N*2 Bytes

Name: BitStream Originator
Type: 7

Identifier:

VCDU_ID	2 Bytes

Parameters:

B_PDU Length in Bytes	4 Bytes

Name: Bitstream Terminator
Type: 8

Identifier:	VCDU_ID	2 Bytes

Parameters: None.

4.5.3 VCA Layer Managed Objects

Name: Outgoing Physical Channel
Type: 9

Identifier:	Physical Channel ID	2 Bytes

Parameters:	VC_PDU Length in Bytes	4 Bytes
	Insert Active	1 Byte
	Insert Size	4 Bytes

The Insert Active Byte has its LSB set to 1 for Insert Active and 0 for Insert Inactive; all other bits are irrelevant.

Name: Incoming Physical Channel
Type: 10

Identifier:	Physical Channel ID	2 Bytes

Parameters:	VC_PDU Length in Bytes	4 Bytes
	Insert Active	1 Byte
	insert Size	4 Bytes

The Insert Active Byte has its LSB set to 1 for Insert Active and 0 for Insert Inactive; all other bits are irrelevant.

Name: VCA_SDU Transmitter
Type: 11

Identifier:	Physical Channel ID	2 Bytes

Parameters:	Options	1 Byte
	Number of Physical Channels	2 Bytes
	Physical Channel IDs	N*2 Bytes

Where the options byte is formatted as follows:

MSB							**LSB**
X	X	X	X	ECF	OCF	HEC	RS

1=Present, 0=Absent, X=Don’t Care

Name: VCA_SDU Receiver
Type: 12

Identifier:

VCDU_ID	2 Bytes

Parameters:

Options	1 Byte
Physical Channel ID	2 Bytes

Where the options byte is formatted as follows:

MSB							**LSB**
X	X	X	X	ECF	OCF	HEC	RS

1=Present, 0=Absent, X=Don’t Care

Name VCDU Transmitter
Type: 13

Identifier:

VCDU_ID	2 Bytes

Parameters:

Number of Physical Channels	2 Bytes
Physical Channel IDs	N*2 Bytes

Name VCDU Receiver
Type: 14

Identifier:

VCDU_ID	2 Bytes

Parameters:

Options	1 Byte
Physical Channel ID	2 Bytes

Where the options byte is formatted as follows:

MSB							**LSB**
X	X	X	X	ECF	OCF	HEC	RS

1=Present, 0=Absent, X=Don’t Care

5 TESTING

As described in section 2, test procedures written in LOTOS were produced and applied to the LOTOS specifications. These tests form an abstract test suite which was implemented and applied to the CCSDS Validation Programme implementations.

Annex A contains descriptions of all these tests and explains how they were used to test the implementations. The test descriptions all conform to a common frame outlined below:

Identifier	A unique identifier will be assigned to each test. Path test identifiers will be prefixed with 'PT'; VCA tests will be prefixed with 'VCAT'; VCLC tests will be prefixed with 'VCLCT'.
Name	Each test will carry a name summarising its purpose.
LOTOS	The implemented tests will be derived from one or more LOTOS tests; cross references will be made to these LOTOS tests.
Aims	A full description of the test will be given.
Addresses	Any addressing information used in the test will be summarised. Logical Data Paths will be referred to by Application Identifier and Qualifier in the form x.y; hence 127.53 refers to a Logical Data Path with an APID of 127 and an APID Qualifier of 53. Virtual Channels will be handled similarly, with 12.35 referring to a VCDU_ID with a Spacecraft Identifier of 12 and a Virtual Channel Identifier of 35. If it is necessary to refer to a particular Packet Channel Identifier within a Virtual Channel, the APID will be prefixed to the VCDU_ID, thus 127.12.35 will refer to Packet Channel 127 within the previously described Virtual Channel.
Inputs	General description of the inputs required.
Outputs	General description of the outputs expected.
Notes	Any supplementary information associated with the test or its execution (e.g., extensions, comments).

In certain cases (such as testing Path's response to the reception, from a subnetwork, of corrupt packets) a special piece of software was required which could inject data units into an underlying Service Provider process. Where the use of such software was necessary, a note will be inserted in the 'Inputs' section of the test description.

All the tests given in Annex A were applied locally to the Validation Programme Implementations. It was intended that the same tests would be used in the interoperability testing exercise; however, because of time and budget restrictions, only a limited amount of interoperability testing was possible involving the ESA and NASA Path protocol implementations and their Management interfaces.

6 RESULTS AND ANALYSIS

The Validation Programme has been largely successful; several errors, inconsistencies and ambiguities have been uncovered in the original issue of the English Language specifications, allowing corrections and clarifications to be made to the first revision of the main Recommendation. Formal specifications of the AOS protocols and services have been produced, along with formally specified test suites. The AOS protocols have been implemented (along with support software) and tested. A limited amount of interoperability testing has been carried out.

The application of LOTOS has proved useful; the act of producing a formal specification forces the specifier/designer to explore the problem completely (even though the specification may be at an abstract level). This benefit manifested itself in the Validation Programme by leading to the identification of the minimum amount of Management Information necessary to support the AOS protocols, as well as by enforcing a complete exploration of the protocol and service operations.

In addition to leading to a complete specification, the use of LOTOS permitted us to rigorously test the operation of the protocols and services without having to write any software, thus problems were identified and corrected at the specification phase. This resulted in a comparatively short implementation phase during which few (if any) errors of specification or design were uncovered.

The LOTOS toolset used within the Validation Programme (Hippo, produced in a collaborative Information Technology Programme run by the European Community) has proved invaluable during the production and testing of the formal specifications. The toolset is, however, limited in scope and usability; several problems have arisen which have resulted in the specifications' being slightly altered in order to conform to the toolset's peculiarities.

The Validation Programme has produced useful information for the CCSDS Management and Signalling Recommendations. The LOTOS specifications of the AOS protocols have identified the minimum Management information required for the correct operation of the protocols, and the Validation Programme implementations have examined cross supported Management Primitives for the ESA and NASA prototype Node Managers.

Finally, the Validation Programme benefited greatly from the high quality of the English Language specifications found in reference [2]. Several problems in the original issue of the main Recommendation were uncovered during the Programme, and Programme-proposed solutions have been incorporated into the current issue of the main Recommendation.

ANNEX A TEST PLAN

A.1 INTRODUCTION

This Annex provides detailed information on the tests and test configurations used during the software implementation phase of the Validation Programme.

A.2 PATH SERVICE TEST DESCRIPTIONS

This section describes the Path Service.

A.2.1 Test Configuration

The basic configuration used for testing the Path implementations is given in Figure A-1 below; this configuration tests both the end-system and relay-system functions of the Path Layer:

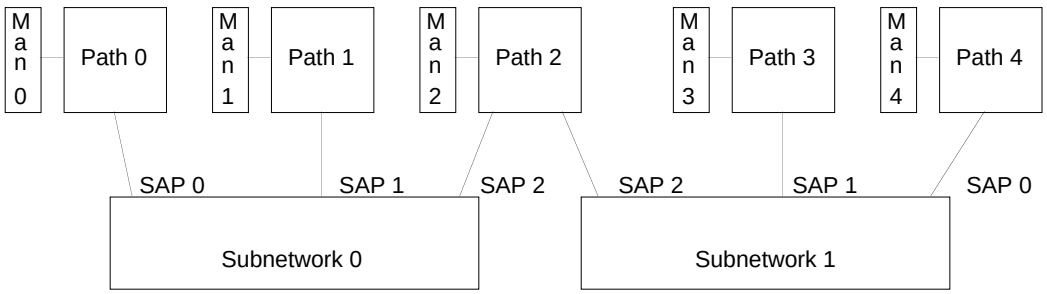


Figure A-1: Basic Configuration for Testing Path Implementations

Of the five Path entities, four are acting as end systems, with Path 2 providing a relaying function between Subnetwork 0 and Subnetwork 1. Note that this configuration is such that the Path must be able to distinguish between identical Subnetwork SAP addresses on different Subnetworks (i.e., the SAPs 0,1 and 2 occur in both Subnetworks).

Eight Logical Data Paths are used. They are shown in the following diagrams.

LDP 1.1 is a straightforward single-end-point Logical Data Path running from Path Entity 2 (Packet Service) and terminating at Path Entity 3 (Packet Service):

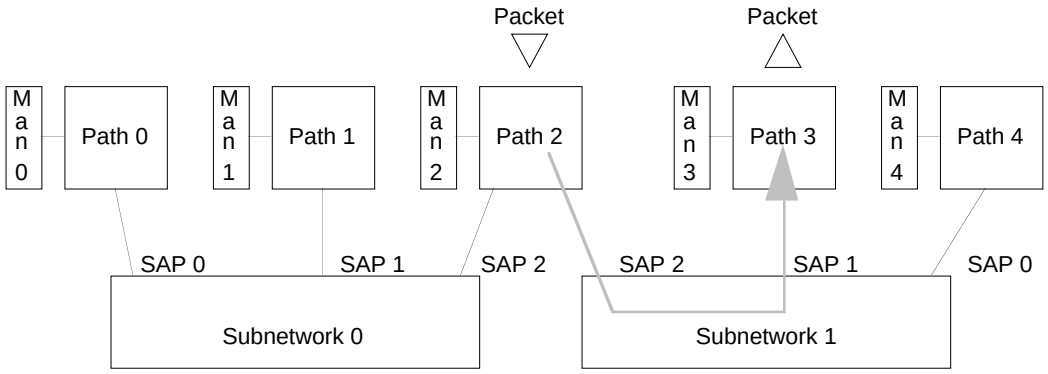


Figure A-2: Logical Data Path 1.1

LDP 11.1 is similar to LDP 1.1, but the terminating service at Path Entity 3 is the Octet String Service:

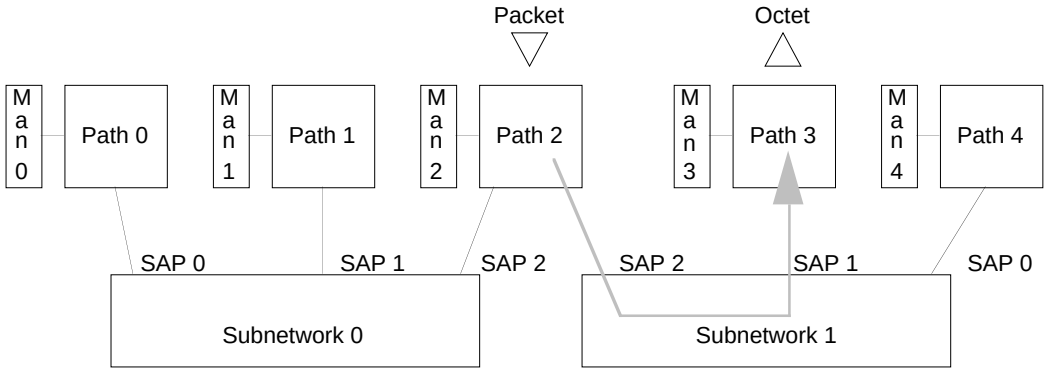


Figure A-3: Logical Data Path 11.1

LDP 2.1 is another straightforward single-end-point Logical Data Path running from Path Entity 2 (Octet String Service) and terminating at Path Entity 0 (Octet String Service):

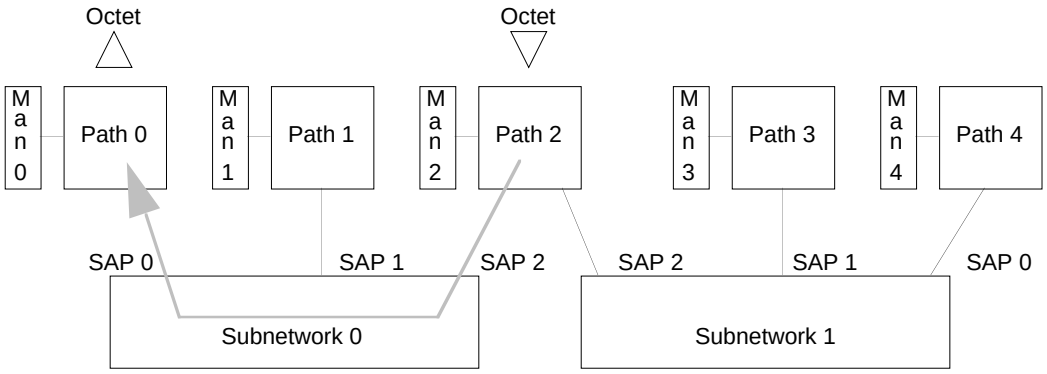


Figure A-4: Logical Data Path 2.1

LDP 22.1 is similar to LDP 2.1, but the terminating service at Path Entity 0 is the Packet Service:

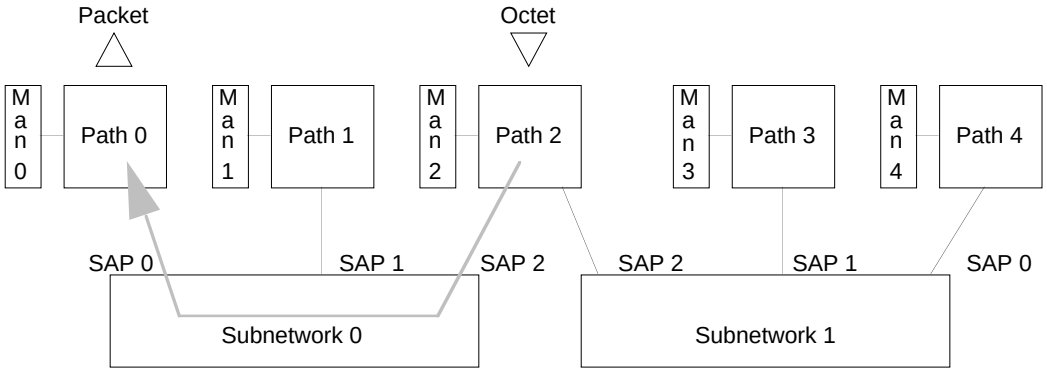


Figure A-5: Logical Data Path 22.1

LDP 3.2 is a slightly more complex Logical Data Path which covers two Subnetworks; it starts at Path Entity 0 (Octet String Service) and terminates at Path Entity 3 (Packet Service):

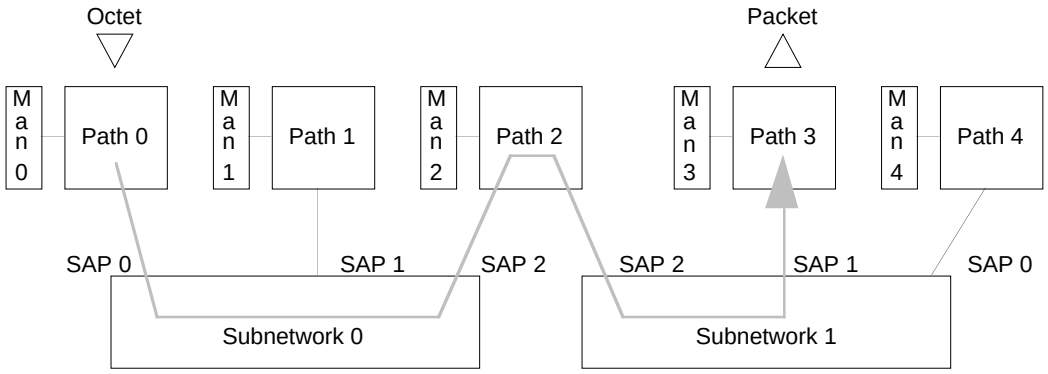


Figure A-6: Logical Data Path 3.2

LDP 4.1 is, again, more complex, being a multiple-end-point Logical Data Path covering two Subnetworks. From its start point at Path Entity 3 (Packet Service), the LDP runs to Path Entity 2 where it is multicast to its three end points. The end points at Path Entities 1 and 4 deliver the Packet Service, the end point at Path Entity 0 delivers the Octet String Service:

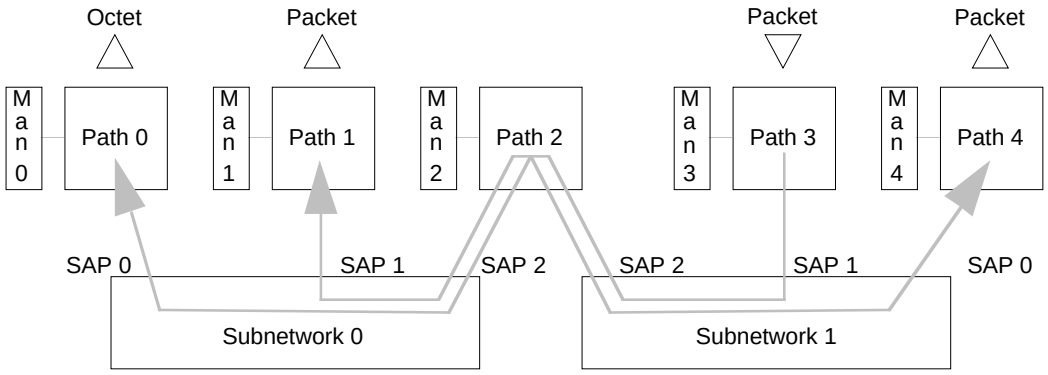


Figure A-7: Logical Data Path 4.1

LDP 3.1 is similar to LDP 4.1, but the originating service at Path Entity 3 is the Octet String Service:

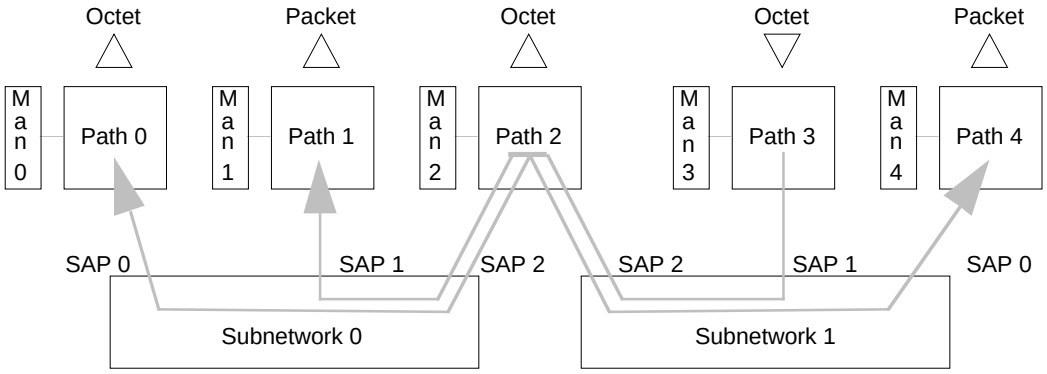


Figure A-8: Logical Data Path 3.1

LDP 5.1 originates at Path Entity 2 (Packet Service) and terminates at Path Entities 0 (Packet Service) and 1 (Octet String Service) on Subnetwork 0, and Path Entities 3 (Packet Service) and 4 (Octet String Service) on Subnetwork 1:

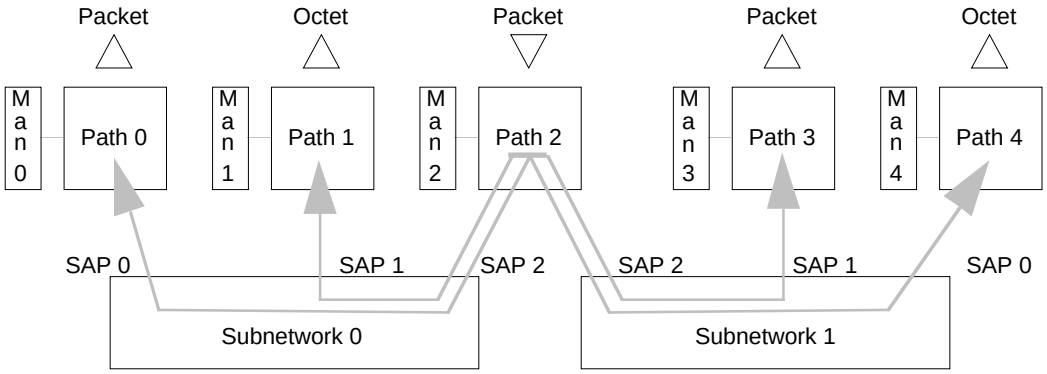


Figure A-9: Logical Data Path 5.1

A.2.2 Path Interoperability Test Configuration

The basic configuration to be used for testing interoperability between Path implementations is given in Figure A-10:

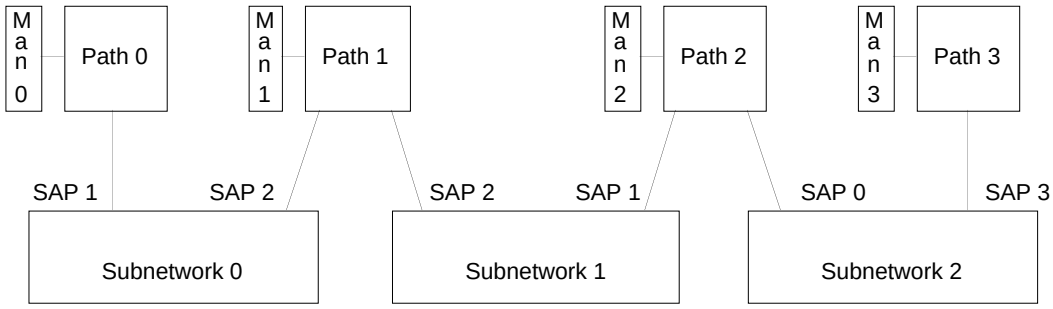


Figure A-10: Basic Configuration for Testing Interoperability between Path Implementations

Of the four Path entities, two are acting as end systems, with Path entity 1 providing a relay function between Subnetwork 0 and Subnetwork 1, and Path entity 2 providing a relay function between Subnetwork 1 and Subnetwork 2. Note that Path Entities 0 and 2 should be provided by one party, and Path Entities 1 and 3 by the other, thus testing both implementations in the end-system and relay-system role.

Three Logical Data Paths are to be used as shown in the following diagrams.

LDP 12.1 is a multiple-end-point Logical Data Path running from Path Entity 0 (Packet Service) to Path Entities 1, 2 (Octet String Service) and 3 (Packet Service):

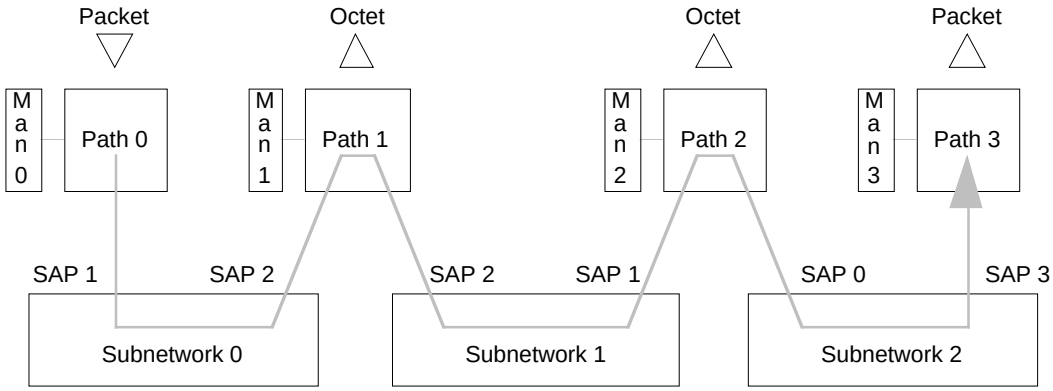


Figure A-11: Logical Data Path 12.1

LDP 2046.12 is similar to the previous Logical Data Path, but it runs in the reverse direction from its origination at Path Entity 3 (Octet String Service) to terminations at Path Entities 2 (Octet String Service), 1 and 0 (Packet Service):

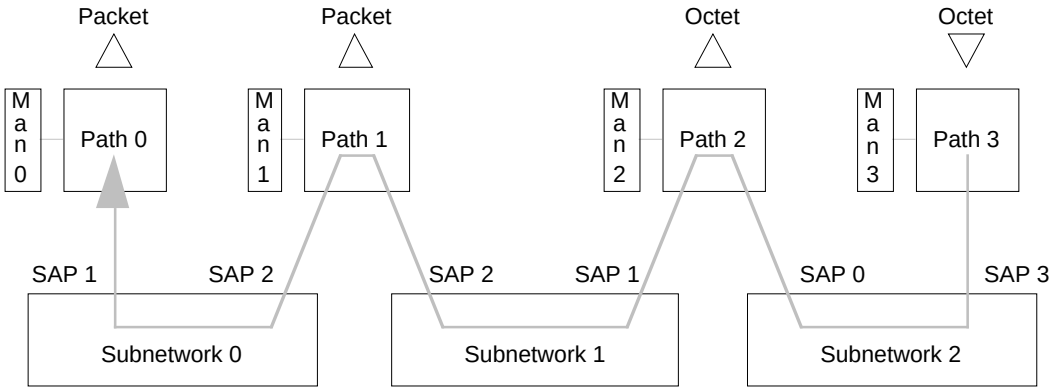


Figure A-12: Logical Data Path 2046.12

LDP 52.52 runs from Path Entity 1, where it is multicast into Subnets 0 and 1. It terminates at Path Entities 0 (Octet String Service), 2 (Packet Service), and 3 (Octet String Service):

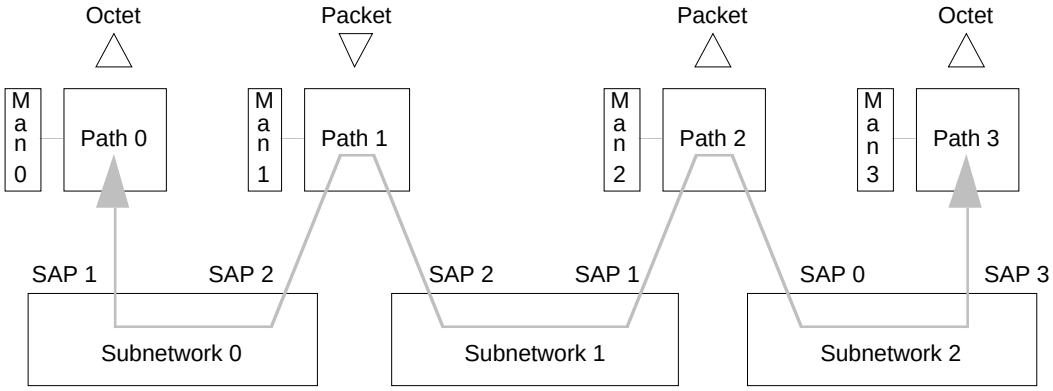


Figure A-13: Logical Data Path 52.52

A.2.3 Path Entity Tests

Identifier	PT1
Name	Service Requests occurring before reception of Management Information.
LOTOS	pp.t1.lot, pp.t1a.lot, ps.t1.lot
Aims	Any Service Requests occurring on Logical Data Paths before Management Information relevant to those Logical Data Paths has been received should be rejected.
Addresses	None.
Inputs	An Octet String Service Request should be made on a Logical Data Path which has not been set up as an Octet String source. A Packet Service Request should be made on a Logical Data Path which has not been set up as a Packet source.
Outputs	The Requests should be discarded and an error message generated.
Notes	None.

A.2.3 Path Entity Tests (continued)

Identifier	PT2
Name	Inappropriate Service Request
LOTOS	pp.t2.lot / ps.t2.lot
Aims	Packet Service Requests should not be allowed over the Octet String Service Interface.
Addresses	LDP 2.1
Inputs	Management Information setting up LDP 2.1 as an Octet String Source. A Packet Service Request should be made on LDP 2.1.
Outputs	The Packet Service Request should be rejected.
Notes	The reverse case (making an Octet String Request on an LDP set up as a Packet Source) should also be exercised.

A.2.3 Path Entity Tests (continued)

Identifier	PT3
Name	Misuse of Path Relayers.
LOTOS	pp.t3.lot / pp.t3a.lot
Aims	Service Requests should not be allowed on LDPs which are set up for relaying purposes only.
Addresses	LDP 3.2
Inputs	Management Information setting up LDP 3.2 as a relay between two Subnetworks. A Packet Service Request should be made on LDP 3.2. An Octet String Service Request should be made on LDP 3.2.
Outputs	Both Service Requests should be rejected.
Notes	None.

A.2.3 Path Entity Tests (continued)

Identifier	PT4
Name	Service Request Checks.
LOTOS	pp.t4.lot, pp.t4a.lot, ps.t3.lot, ps.t3a.lot
Aims	Packets received in Service Requests should be checked to ensure that they obey certain restrictions.
Addresses	LDP 1.1
Inputs	A Packet Service Request where the APID is a reserved APID. A Packet Service Request where the Version field is not set to Version 1. A Packet Service Request where the Packet Length Field does not agree with the length of the User Data. A Packet Service Request where the length of the packet exceeds the maximum SDU length set by Management. A series of Packet Service Requests containing valid Packets.
Outputs	The Packet Service Requests containing invalid Packets should be discarded with error messages being generated. The Packet Service Request containing valid Packets should be accepted and Subnetwork Service Requests generated.
Notes	None.

A.2.3 Path Entity Tests (continued)

Identifier	PT5
Name	Service Indication Checks
LOTOS	pp.t5.lot, pp.t5a.lot
Aims	Packets received in Service Indications should be checked to ensure that they obey certain restrictions.
Addresses	LDP 1.1
Inputs	A Subnetwork Service Indication where the APID is a reserved APID. A Subnetwork Service Indication on an unconfigured Subnetwork. A Subnetwork Service Indication where the Version field is not set to Version 1. A Subnetwork Service Indication where the Packet Length Field does not agree with the length of the User Data. A Subnetwork Service Indication on an unconfigured Subnetwork SAP. A Subnetwork Service Indication where the length of the packet exceeds the maximum SDU length set by Management. A series of Subnetwork Service Indications containing valid Packets.
Outputs	The Subnetwork Service Indications containing invalid Packets should be discarded with error messages being generated. The Subnetwork Service Indications containing valid Packets should be accepted and Packet Service Indications generated.
Notes	None.

A.2.3 Path Entity Tests (continued)

Identifier	PT6
Name	Service Indication Checks.
LOTOS	pp.t6.lot
Aims	Packets received in Service Indications should be checked to ensure that they obey certain restrictions.
Addresses	LDP 11.1
Inputs	A Subnetwork Service Indication where the APID is a reserved APID. A Subnetwork Service Indication where the Version field is not set to Version 1. A Subnetwork Service Indication where the Packet Length Field does not agree with the length of the User Data. A Subnetwork Service Indication on an unconfigured Subnetwork SAP. A series of Subnetwork Service Indications containing valid Packets.
Outputs	The Subnetwork Service Indications containing invalid Packets should be discarded with error messages being generated. The Subnetwork Service Indications containing valid Packets should be accepted and Octet String Service Indications generated.
Notes	None.

A.2.3 Path Entity Tests (continued)

Identifier	PT7
Name	Single Relay.
LOTOS	pp.t7.lot
Aims	Once an LDP relay Managed Object has been activated, the Path should relay Packets received from the Subnetwork.
Addresses	LDP 3.1
Inputs.	Management Information setting up an LDP relayer, accepting Packets from the Subnetwork and relaying them to a local and a remote User Entity. A Subnetwork Service Indication containing a valid Packet.
Outputs	The Packet contained in the Subnetwork Indication should emerge in a locally generated Octet String Service Indication and in a Subnetwork Service Request.
Notes	None.

A.2.3 Path Entity Tests (continued)

Identifier	PT8
Name	Multiple Relays.
LOTOS	pp.t8.lot
Aims	Several relays may be set up operating on different LDPs; they should not interfere with one another.
Addresses	LDP3.1, LDP3.2
Inputs	Management Information setting up the LDP relayers, accepting Packets from the Subnetwork and relaying them to local and remote User Entities. A series of Subnetwork Service Indication containing valid Packets.
Outputs	The Packets contained in the Subnetwork Indications should emerge in locally generated Octet String Service Indication and in a Subnetwork Service Request.
Notes	None.

A.2.3 Path Entity Tests (continued)

Identifier	PT9
Name	Multicasting.
LOTOS	pp.t9.lot. ps.t4.lot, ps.t6.lot
Aims	An LDP Relayer should be capable of generating several Subnetwork Service Requests in response to a single Path Service Request.
Addresses	LDP 5.1
Inputs	A single Packet Service Request initiated on Path Entity 2.
Outputs	Four Subnetwork Service Requests should be issued from Path Entity 2, resulting in Packet Indications at Entities 0 and 3, and Octet String Indications at Entities 1 and 4.
Notes	None.

A.2.3 Path Entity Tests (continued)

Identifier	PT10
Name	Multicasting.
LOTOS	pp.t10.lot
Aims	An LDP Relayer should be capable of generating several Subnetwork Service Requests in response to a single Subnetwork Service Indication.
Addresses	LDP4.1
Inputs	A single Subnetwork Service Request should be injected into Subnetwork 1 using the tester process. The Subnetwork Request should have a source Subnetwork SAP of 1 and a destination Subnetwork SAP of 2.
Outputs	Three Subnetwork Service Requests should be issued from Path Entity 2, resulting in Packet Indications at Entities 1 and 4, and an Octet String Indication at Entity 0.
Notes	None.

A.2.3 Path Entity Tests (continued)

Identifier	PT11
Name	Path Request Checks.
LOTOS	pp.t11.lot, ps.t5.lot, ps.t5a.lot
Aims	Octet String Service Requests should be checked to ensure that they obey certain restrictions.
Addresses	LDP22.1
Inputs	An Octet String Service Request where the SDU exceeds the maximum size. An Octet String Service Request where the PathID is unknown. Several correctly formatted Octet String Service Requests, some with Secondary Header Indicators set to True, others to False.
Outputs	The first two Octet String Service Requests should be discarded with error messages generated. The correctly formatted Octet String Service Requests should be accepted, and Subnetwork Service Requests generated which contain CCSDS Version-1 Packets formatted as required (e.g., with sequentially increasing Sequence Counts and with the Secondary Header Flag field set according to the Secondary Header Indicators).
Notes	None.

A.2.3 Path Entity Tests (continued)

Identifier	PT12
Name	Data Loss Flags with Packet Repetition.
LOTOS	pp.t12.lot
Aims	When Data Loss Flags are generated, and the Path detects discontinuities in the Packets arriving on the associated Logical Data Path, Data Loss should be signalled to the User Entity.
Addresses	LDP11.1
Inputs	Four Subnetwork Service Requests should be injected into Subnetwork 1 using the tester process. The Subnetwork Requests should have a source Subnetwork SAP of 2 and a destination Subnetwork SAP of 1. The Packets contained in the Requests should have their Packet Sequence Count fields set to 0, 1, 1, 2 in that sequence.
Outputs	The Octet String Service Indications generated at Path Entity 3 should have Data Loss Flags set to False, False, True and False in that sequence.
Notes	None.

A.2.3 Path Entity Tests (continued)

Identifier	PT13
Name	Data Loss Flags with Packet Disorder.
LOTOS	pp.t13.lot
Aims	When Data Loss Flags are generated, and the Path detects discontinuities in the Packets arriving on the associated Logical Data Path, Data Loss should be signalled to the User Entity.
Addresses	LDP11.1
Inputs	Six Subnetwork Service Requests should be injected into Subnetwork 1 using the tester process. The Subnetwork Requests should have a source Subnetwork SAP of 2 and a destination Subnetwork SAP of 1. The Packets contained in the Requests should have their Packet Sequence Count fields set to 0, 1, 3, 2, 4, 5 in that sequence.
Outputs	The Octet String Service Indications generated at Path Entity 3 should have Data Loss Flags set to False, False, True, True, True and False in that sequence.
Notes	None.

A.2.3 Path Entity Tests (continued)

Identifier	PT14
Name	Multiple Octet String Service Requests.
LOTOS	pp.t14.lot, pp.t16.lot, ps.t7.lot
Aims	The Path should handle multiple Octet String Service Requests, generating the correct Sequence Counts and default Packet field values.
Addresses	LDP22.1
Inputs	A number of Octet String Service Requests (at least 16,387) should be made at Path Entity 2.
Outputs	The same number of Packet Service Indications should be generated at Path Entity 0, with the Packet Sequence Count wrapping from 16,383 to 0, the Packet APID field being set to 22, the Packet Sequence Flags being set to UnSegmented, the Packet Version being set to Version1, and the Packet Length field being set to correctly indicate the length of the User Data passed in the Octet String Service Requests.
Notes	None.

A.2.3 Path Entity Tests (continued)

Identifier	PT15
Name	Multiple Packet Service Requests.
LOTOS	pp.t15.lot, ps.t8.lot
Aims	The Path should handle multiple Packet Service Requests.
Addresses	LDP1.1
Inputs	A number of Packet Service Requests (at least 16,387) should be made at Path Entity 2.
Outputs	The same number of Packet Service Indications should be generated at Path Entity 3, with the Packets contained in the Indications having the same values as those set in the Requests.
Notes	None.

A.2.3 Path Entity Tests (continued)

Identifier	PT16
Name	Multiple Subnetwork Service Indications.
LOTOS	pp.t16.lot
Aims	The Path should handle multiple Subnetwork Service Indications.
Addresses	LDP1.1
Inputs	A number of Subnetwork Service Requests (at least 16,387) should be injected into Subnetwork 1 using the tester process. The Subnetwork Service Requests should have a source Subnetwork SAP of 2 and a destination Subnetwork SAP of 1.
Outputs	The same number of Packet Service Indications should be generated at Path Entity 3, with the Packets contained in the Indications having the same values as those set in the Subnetwork Service Requests.
Notes	None.

A.3 SPACE LINK SUBNETWORK SERVICE TESTS

This section covers the VCA and VCLC testing.

A.3.1 VCA Service Tests

This section covers the VCA service.

A.3.2 Test Configuration

The basic configuration used for VCA Entity testing is shown below:

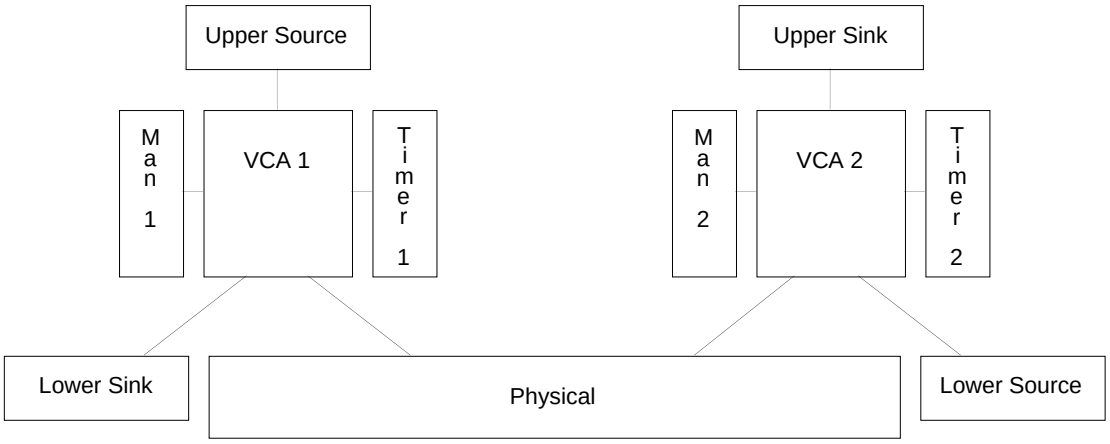


Figure A-14: Basic Configuration for VCA Entity Testing

The configuration consists of two VCA Entities connected by a Physical Channel Process. Eleven Virtual Channels will be used in total; their characteristics are outlined in Table A-1.

Table A-1: Virtual Channel Characteristics

VCDU_ID[1]		Direction	Service	Tests
SCID	VCID			
5	3	2 -> 1	UNITDATA	1, 5, 6
5	4	1 -> 2	UNITDATA	3
6	1	2 ->1	UNITDATA	2
6	2	2 ->1	UNITDATA	2
6	3	2 ->1	UNITDATA	2
6	4	2 ->1	UNITDATA	2
6	5	2 ->1	UNITDATA	2
6	6	2 ->1	UNITDATA	2
6	7	2 ->1	UNITDATA	2
6	8	2 ->1	UNITDATA	2
5	30	2 ->1	VCDU	4, 5

[1]VCDU_ID = Virtual Channel Data Unit Identifier, a concatenation of the Spacecraft Identifier (SCID) and Virtual Channel Identifier (VCID)

A.3.3 VCA Entity Tests

Identifier	VCAT1
Name	VCA_SDU Length Check.
LOTOS	vcap.t1.lot
Aims	The Channel Access Data Units (CADUs) for a Physical Channel must be of a fixed length over the lifetime of that channel. The VCA Sublayer must, therefore, only accepts VCA_SDUs which exactly fit the data zone of the VCDU/CVCDU to be transmitted over the Physical Channel in question.
Addresses	VCDU_ID 5.3
Inputs	A VCA_UNITDATA Service Request containing a VCA_SDU which is too large to be inserted into the VC_PDU data zone. A VCA_UNITDATA Service Request containing a VCA_SDU which is too small to be inserted into the VC_PDU data zone. A VCA_UNITDATA Service Request containing a VCA_SDU which exactly fits the VC_PDU data zone.
Outputs	The first and second VCA_UNITDATA Service Requests should be rejected, and the third accepted.
Notes	No output should be produced by the VCA Sublayer until it is directed to send a CADU by the timer.

A.3.3 VCA Entity Tests (continued)

Identifier	VCAT2
Name	VCDU/CVCDU Format.
LOTOS	vcap.t2.lot
Aims	Each VCDU_ID has associated with it a number of parameters defining the presence or absence of Reed-Solomon Check Symbols, the presence or absence of the VCDU Header Error Control Field (shortened Reed-Solomon), and the presence or absence of the VCDU Trailer Field (and its contents). Each of these parameters affect the VCDU/CVCDU format.
Addresses	VCDU_IDs 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 6.8
Inputs	Eight VCA_UNITDATA Service Requests, each containing a VCA_SDU of the correct length for each VCDU_ID. Each VCA_UNITDATA Service Request should be followed by a transmission timer event to cause transmission of the generated CADU.
Outputs	Eight CADUs should be generated containing VC_PDUs on the following VCDU_IDs with the stated configuration of optional fields: 1. VCDU_ID 6.1 Reed-Solomon present 2. VCDU_ID 6.2 Reed-Solomon Absent 3. VCDU_ID 6.3 VCDU Header Error Control present 4. VCDU_ID 6.4 VCDU Header Error Control absent 5. VCDU_ID 6.5 VCDU Trailer Operational Control Field present 6. VCDU_ID 6.6 VCDU Trailer Operational Control Field absent 7. VCDU_ID 6.7 VCDU Trailer Error Control Field present 8. VCDU_ID 6.8 VCDU Trailer Error Control Field absent
Notes	Note that some of the above combinations are illegal in terms of Management, e.g., the case where only the VCDU Trailer Operational Control Field is present. This test is only intended to exercise the VCA Sublayer in terms of producing optional VCDU/CVCDU fields.

A.3.3 VCA Entity Tests (continued)

Identifier	VCAT3
Name	VC_PDU Discontinuities.
LOTOS	vcap.t3.lot
Aims	The VCA Sublayer should, if requested, generate VCDU Data Loss Flags derived from the VCDU/CVCDU counter field.
Addresses	VCDU_ID 5.4
Inputs	Five complete CADUs should be passed to the VCA Sublayer in Physical Service Indications. The five VC_PDUs contained in the CADUs should have counter fields set to X, X+2, X+4, X+5, and X+7 in that order.
Outputs	Five VCA_UNITDATA Service Indications should be generated with the Data Loss Flags being set to False, True, True, False and True in that order.
Notes	None.

A.3.3 VCA Entity Tests (continued)

Identifier	VCAT4
Name	Inappropriate Address.
LOTOS	vcap.t4.lot, vcap.t5.lot
Aims	Each VCDU_ID is associated with an upper layer service by Management. Use of the VCA_UNITDATA Service should not be allowed on a VCDU_ID which is not associated with that service.
Addresses	VCDU_ID 5.30
Inputs	A VCA_UNITDATA Service Request on VCDU_ID 5.30 A VCA_UNITDATA Service Request on VCDU_ID 6.30
Outputs	Both VCA_UNITDATA Service Requests should be rejected.
Notes	Two conditions are tested here; incorrect use of a known SAP (5.30) and attempted use of an unknown SAP (6.30).

A.3.3 VCA Entity Tests (continued)

Identifier	VCAT5
Name	Correct function of Release Parameters.
LOTOS	vcap.t8.lot
Aims	Each VCDU_ID has associated with it parameters which define when the VC_PDU must be released for transmission.
Addresses	VCDU_IDs 5.3, 5.30
Inputs	Two VCA_UNITDATA Service Requests on VCDU_ID 5.3 A VCDU Service Request on VCDU_ID 5.30 Three VCDU Release Events on the required Physical Channel.
Outputs	Three CADUs should be generated containing the VC_PDUs in an order determined by the project specified Release Parameters.
Notes	The ESA VCA Sublayer implementation provides a simple FIFO Release Algorithm. This test is still useful, however, as it exercises the release timer which governs the transmission of CADUs.

A.3.3 VCA Entity Tests (continued)

Identifier	VCAT6
Name	Inappropriate Address.
LOTOS	vcap.t9.lot, vcap.t10.lot
Aims	Each VCDU_ID is associated with an upper layer service by Management. Use of the VCDU Service should not be allowed on a VCDU_ID which is not associated with that service.
Addresses	VCDU_ID 5.3
Inputs	A VCA_VCDU Service Request on VCDU_ID 5.3 A VCA_VCDU Service Request on VCDU_ID 6.30
Outputs	Both VCA_VCDU Service Requests should be rejected.
Notes	Two conditions are tested here; incorrect use of a known SAP (5.3) and attempted use of an unknown SAP (6.30).

A.3.4 VCLC Service Tests

This section covers the VCLC service.

A.3.5 Test Configuration

The basic configuration used for VCLC Entity testing is shown in Figure A-15.

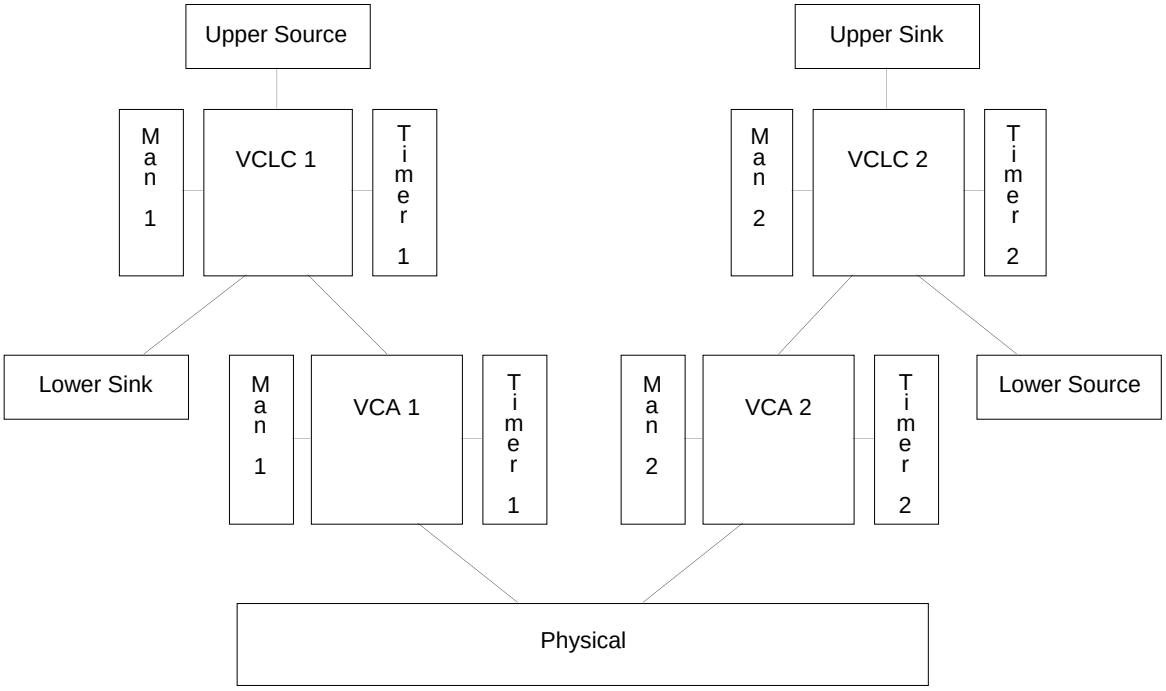


Figure A-15: Basic Configuration for VCLC Entity Testing

The configuration consists of two VCLC Entities connected by two VCA Entities and a Physical Channel Process. Three Virtual Channels will be used in total; their characteristics are outlined in Table A-2.

Table A-2: Virtual Channel Characteristics

VCDU_ID[2]		PCID[3]	Direction	Service	Tests
SCID	VCID				
5	3	2046	2 -> 1	ENCAPSULATION	1, 10
5	3		2 -> 1	MULTIPLEXING	1, 3, 4, 5, 7, 10, 12, 17
5	4	2046	1 -> 2	ENCAPSULATION	2, 6, 9, 11, 13
5	4		1 -> 2	MULTIPLEXING	2, 6, 7, 9, 11, 13
5	1		2 -> 1	BITSTREAM	8, 14
5	2		1 -> 2	BITSTREAM	15, 16

2VCDU_ID = Virtual Channel Data Unit Identifier, a concatenation of the Spacecraft Identifier (SCID) and Virtual Channel Identifier (VCID)

3PCID = Packet Channel Identifier (equivalent to the Application Process Identifier or APID).

A.3.6 VCLC Entity Tests

Identifier	VCLCT1
Name	Multiplexing.
LOTOS	vclcp.t1.lot
Aims	The VCLC Sublayer should multiplex together M_SDUs and E_PDUs (both Version-1 CCSDS Packets), producing M_PDUs which will form part of a VCA_UNITDATA Service Request.
Addresses	VCDU_ID 5.3, PCID 2046
Inputs	Two M_UNITDATA Service Requests, in which the CCSDS Packets are sized such that together they exceed the Packet Zone size set up for the Multiplexing procedures. An E_UNITDATA Service Request which (together with the remainder of the second packet above) will produce a packet that fills the Packet Zone of the next M_PDU.
Outputs	Two VCA_UNITDATA Service Requests should be produced. The first should contain one complete packet from the first M_UNITDATA Service Request and a partial packet from the second M_UNITDATA Service Request. The second should contain the remainder of the packet from the second M_UNITDATA Service Request and a complete packet resulting from the E_UNITDATA Service Request.
Notes	None.

A.3.6 VCLC Entity Tests (continued)

Identifier	VCLCT2
Name	Demultiplexing.
LOTOS	vclcp.t2.lot
Aims	The VCLC Sublayer should demultiplex E_PDUs and M_SDUs from the M_PDUs, generating M_UNITDATA Service Indications and E_UNITDATA Service Indications.
Addresses	VCDU_ID 5.4, PCID 2046
Inputs	A VCA_UNITDATA Service Indication containing an M_PDU with the FHP set to zero, containing one complete and one partial packet each with user APIDs. A VCA_UNITDATA Service Indication containing an M_PDU with the FHP set to point to the first packet header in the M_PDU, containing one partial packet and one complete packet; the partial packet being the completion of the segmented packet above, the complete packet having an APID of 2046.
Outputs	Two M_UNITDATA Service Indications should be generated containing the first two packets referred to above. One E_UNITDATA Service Indication should be generated containing the user data from the third packet.
Notes	None.

A.3.6 VCLC Entity Tests (continued)

Identifier	VCLCT3
Name	Correct function of the First Header Pointer.
LOTOS	vclcp.t3.lot
Aims	The First Header Pointer (FHP) placed into M_PDUs by the VCLC Sub-layer should facilitate delimitation of the variable length SDUs contained within the Packet Zones of each M_PDU.
Addresses	VCDU_ID 5.3
Inputs	Several M_UNITDATA Service Requests should be made such that a CCSDS Packet Header is split between two M_PDUs, where the second M_PDU contains another complete packet header. Several M_UNITDATA Service Requests should be made such that an M_PDU is produced which contains only packet data, i.e., no CCSDS Packet Header. A Release Parameter Event should be generated such that an M_PDU is produced containing only a CCSDS Fill Packet.
Outputs	The first M_UNITDATA Service Requests should produce two VCA_UNITDATA Service Requests. The first Request should contain an M_PDU with its First Header Pointer pointing to the first header in the packet zone. The second should contain an M_PDU with its First Header Pointer pointing to the complete packet header (ignoring the fractional continuation from the previous M_PDU). The next batch of M_UNITDATA Service Requests should produce another VCA_UNITDATA Service Request containing an M_PDU with its First Header Pointer set to ‘all ones’ (to signify that there are no packet headers present). The Release Parameter Event should produce another VCA_UNITDATA Service Request containing an M_PDU with its First Header Pointer set to ‘all ones minus one’ (signifying that the M_PDU contains fill data).
Notes	None.

A.3.6 VCLC Entity Tests (continued)

Identifier	VCLCT4
Name	Correct function of the Release Parameters.
LOTOS	vclcp.t4.lot
Aims	Each VCDU_ID may have parameters associated with it which determine when data units should be released.
Addresses	VCDU_ID 5.3
Inputs	Several M_UNITDATA Service Requests such that the M_PDU Packet Zone is not completely filled. These Requests should be followed by a Release Parameter Event on VCDU_ID 5.3.
Outputs	A VCA_UNITDATA Service Request should be generated containing a filled M_PDU where the CCSDS Fill Packet has an APID of ‘all ones’.
Notes	None.

A.3.6 VCLC Entity Tests (continued)

Identifier	VCLCT5
Name	VCA_SDU Length Check.
LOTOS	vclcp.t5.lot
Aims	The Channel Access Data Units (CADUs) for a Physical Channel must be of a fixed length over the lifetime of that channel. The VCLC Sublayer must, therefore, produce VCA_SDUs which exactly fit the data zone of the VCDU/CVCDU to be transmitted over the Physical Channel in question.
Addresses	VCDU_ID 5.3
Inputs	Several M_UNITDATA Service Requests such that their overall length exceeds the length of the Packet Zone of the M_PDU.
Outputs	A VCA_UNITDATA Service Request should be generated, where the VCA_SDU is of the required length.
Notes	None.

A.3.6 VCLC Entity Tests (continued)

Identifier	VCLCT6
Name	Packet Channel ID Checks.
LOTOS	vclcp.t6.lot, vclcp.t11.lot
Aims	The VCLC Sublayer should check the APID (PCID) fields of the packets contained in the M_PDUs received from the VCA Sublayer.
Addresses	VCDU_ID 5.4, PCIDs 34 and 2046
Inputs	A VCA_UNITDATA Service Indication containing four complete packets, the first of which should have an APID of 34 (i.e., a user APID), the second should have an APID of 2046 (i.e., the 8473 Encapsulation APID), the third should have an APID of 2045 (i.e., an unassigned reserved APID), and the last packet should have an APID of 2047 (i.e., the Fill APID).
Outputs	A single M_UNITDATA Service Indication should be generated containing the packet with an APID of 34; a single E_UNITDATA Service Indication should be generated containing the user data from the packet with an APID of 2046. The two other packets should be discarded.
Notes	None.

A.3.6 VCLC Entity Tests (continued)

Identifier	VCLCT7
Name	Non-Version1 CCSDS Packets.
LOTOS	vclcp.t7.lot
Aims	The VCLC Sublayer should not accept any M_SDUs (i.e., CCSDS Packets) which are not formatted as Version-1 Packets.
Addresses	VCDU_IDs 5.4 and 5.3
Inputs	Two M_UNITDATA Service Requests should be made containing packets of such a size that they completely fill the M_PDU Packet Zone. The first Request should contain a non-Version1 Packet, the second should contain a Version1 Packet. These Requests should be followed by a Release Parameter Event on VCDU_ID 5.3. A VCA_UNITDATA Service Indication should be generated with an M_PDU containing two complete packets, one of which should be a non-Version1 Packet, the other a Version1 Packet.
Outputs	The Release Parameter Event should cause a VCA_UNITDATA Service Request to be generated where the VCA_SDU contains the Version1 Packet and a Fill Packet. The VCA_UNITDATA Service Indication should cause an M_UNITDATA Service Indication to be generated, containing the Version1 Packet.
Notes	None.

A.3.6 VCLC Entity Tests (continued)

Identifier	VCLCT8
Name	Inappropriate VCDU_ID.
LOTOS	vclcp.t8.lot, vclcp.t9.lot, vclcp.t10.lot
Aims	Each VCDU_ID is associated with an upper layer service; use of the Multiplexing Service should not be allowed on a VCDU_ID which is not associated with that Service.
Addresses	VCDU_ID 5.1
Inputs	An M_UNITDATA Service Request should be made on VCDU_ID 5.1 An M_UNITDATA Service Request should be made on VCDU_ID 20.12
Outputs	Both M_UNITDATA Service Requests should be rejected.
Notes	Two conditions are tested here; incorrect use of a known SAP (5.1), and attempted use of an unknown SAP (20.12).

A.3.6 VCLC Entity Tests (continued)

Identifier	VCLCT9
Name	Defective FHP or M_PDU Packet Zone.
LOTOS	vclcp.t12.lot
Aims	When the FHP and Packet Length fields disagree within an M_PDU, the VCLC Sublayer should attempt to continue demultiplexing using the FHP indicated length.
Addresses	VCDU_ID 5.4, PCID 2046
Inputs	Two VCA_UNITDATA Service Indications should be generated. The M_PDU in the first Indication should contain one complete and one partial packet (both on user APIDs), with the FHP set (correctly) to zero, the packet length field of the second packet should be set incorrectly. The M_PDU in the second Indication should contain one partial and one complete packet (the complete packet having an APID of 2046). The FHP should be set correctly so that it will disagree with the packet length field of the partial packet.
Outputs	One M_UNITDATA Service Indication should be generated, containing the first packet, the second packet should be discarded, and an E_UNITDATA Service Indication should be generated containing the user data from the third packet.
Notes	None.

A.3.6 VCLC Entity Tests (continued)

Identifier	VCLCT10
Name	Encapsulation.
LOTOS	vclcp.t13.lot
Aims	The VCLC Sublayer should encapsulate E_SDUs (Octet Strings) in correctly formatted CCSDS Version-1 Packets.
Addresses	VCDU_ID 5.3, PCID 2046
Inputs	Five E_UNITDATA Service Requests should be made, arranged such that they will lead to the production of one VCA_UNITDATA Service Request.
Outputs	A VCA_UNITDATA Service Request should be generated, the M_PDU within which should contain five Version-1 CCSDS Packets, with the Version fields set to (0,0,0), the Secondary Header Flags set to (0), the APIDs set to 2046, the Sequence Counts set sequentially and the Sequence Flags set to (1,1); the Packet Length fields and the user data fields will, of course, depend on the contents of the E_SDUs.
Notes	None.

A.3.6 VCLC Entity Tests (continued)

Identifier	VCLCT11
Name	De-Encapsulation.
LOTOS	vclcp.t14.lot
Aims	The VCLC Sublayer should remove the Packet Headers from the E_PDUs received as a result of VCA_UNITDATA Service Indications. The User Data fields should be sent, unchanged, to the Encapsulation Service User.
Addresses	VCDU_ID 5.4, PCID 2046
Inputs	A single VCA_UNITDATA Service Indication should be generated, containing five complete Version-1 CCSDS Packets, all with APIDs of 2046.
Outputs	Five E_UNITDATA Service Indications should be generated, containing the User Data fields from the five CCSDS Packets.
Notes	None.

A.3.6 VCLC Entity Tests (continued)

Identifier	VCLCT12
Name	Inappropriate address.
LOTOS	vclcp.t15.lot, vclcp.t18.lot
Aims	Each VCDU_ID is associated with an upper layer service by Management; use of the E_UNITDATA Service should not be allowed on a VCDU_ID which is not associated with the M_UNITDATA and E_UNITDATA Services.
Addresses	VCDU_ID 5.3
Inputs	An E_UNITDATA Service Request should be made on VCDU_ID 5.3 An E_UNITDATA Service Request should be made on VCDU_ID 20.12
Outputs	Both E_UNITDATA Service Requests should be rejected.
Notes	Two conditions are tested here; incorrect use of a known SAP and attempted use of an unknown SAP.

A.3.6 VCLC Entity Tests (continued)

Identifier	VCLCT13
Name	E_PDU Discontinuities.
LOTOS	vclcp.t19.lot
Aims	The VCLC Sublayer should, if requested, generate E_SDU Data Loss Flags derived from the Sequence Count fields of successive E_PDUs.
Addresses	VCDU_ID 5.4, PCID 2046
Inputs	A single VCA_UNITDATA Service Indication should be generated, containing five complete Version-1 CCSDS Packets, all with APIDs of 2046. The Sequence Count fields should be set to X, X+2, X+4, X+5, and X+7 in that order.
Outputs	Five E_UNITDATA Service Indications should be generated, with Data Loss Flags set to False, True, True, False and True in that order.
Notes	None.

A.3.6 VCLC Entity Tests (continued)

Identifier	VCLCT14
Name	B_PDU Construction.
LOTOS	vclcp.t20.lot, vclcp.t22.lot, vclcp.t23.lot
Aims	The VCLC Sublayer should correctly format the B_PDU Data Pointer and Data Field.
Addresses	VCDU_ID 5.1
Inputs	Several BITSTREAM Service Requests should be made such that the data exactly fill a B_PDU Bitstream Data Zone. Several BITSTREAM Service Request should be made such that the data do not fill a B_PDU Bitstream Data Zone, these Requests should be followed by a Release Parameter Event. A Release Parameter Event should be generated such that a B_PDU is produced containing only fill data.
Outputs	Three VCA_UNITDATA Service Requests should be generated. The first should contain a B_PDU with the Bitstream Data Pointer set to ‘all ones’. The second should contain a B_PDU with the Bitstream Data Pointer set to point to the number of the last valid user data bit-1 (i.e., if there are eight valid data bits, the BDP should be set to seven). The third request should contain a B_PDU with the BDP set to ‘all ones minus one’.
Notes	None.

A.3.6 VCLC Entity Tests (continued)

Identifier	VCLCT15
Name	B_SDU Extraction.
LOTOS	vclcp.t21.lot
Aims	The VCLC Sublayer should extract the B_SDUs from VCA_SDUs received and send them to the BITSTREAM Service User.
Addresses	VCDU_ID 5.2
Inputs	A single VCA_UNITDATA Service Indication should be generated containing a B_PDU.
Outputs	One or more BITSTREAM Service Indications should be generated (the exact number will depend on the size of the B_SDU selected for the particular implementation). Three VCA_UNITDATA Service Requests should be generated. The Indications should contain the Bitstream data from the received B_PDU in the same bit order.
Notes	None.

A.3.6 VCLC Entity Tests (continued)

Identifier	VCLCT16
Name	B_SDU Extraction with Data Loss
LOTOS	vclcp.t24.lot, vclcp.t25.lot
Aims	The VCLC Sublayer should, if requested, generate Data Loss Flags derived from the Data Loss Flags received in VCA_UNITDATA Service Indications.
Addresses	VCDU_ID 5.2
Inputs	Five VCA_UNITDATA Service Indications should be generated with Data Loss Flags set to False, True, True, False and True in that order.
Outputs	The length of the B_SDUs generated is not specified by CCSDS. If the B_SDUs generated correspond one-to-one with the Bitstream Data Zones, then the Data Loss Flags delivered to the BITSTREAM Service User should be exactly the same as those passed by the VCA Sublayer. If two or more B_SDUs are generated for each Bitstream Data Zone, then only the first BITSTREAM Service Indication should be accompanied by a Data Loss Flag set to the value of the VCA_UNITDATA Service Indication's Data Loss Flag; subsequent BITSTREAM Service Indications should be accompanied by Data Loss Flags set to the value False.
Notes	None.

A.3.6 VCLC Entity Tests (continued)

Identifier	VCLCT17
Name	Inappropriate address
LOTOS	
Aims	Each VCDU_ID is associated with an upper layer service by Management. Use of the BITSTREAM Service should not be allowed on a VCDU_ID which is not associated with that Service.
Addresses	VCDU_ID 5.3
Inputs	A BITSTREAM Service Request should be made on VCDU_ID 5.3 A BITSTREAM Service Request should be made on VCDU_ID 20.12
Outputs	Both BITSTREAM Service Requests should be rejected.
Notes	Two conditions are tested here; incorrect use of a known SAP and attempted use of an unknown SAP.